Software effort interval prediction via Bayesian inference and synthetic Bootstrap resampling

Song, Liyan; Minku, Leandro L.; Yao, Xin

DOI:
10.1145/3295700

Document Version
Peer reviewed version

Citation for published version (Harvard):
Song, L, Minku, LL & Yao, X 2019, 'Software effort interval prediction via Bayesian inference and synthetic Bootstrap resampling', *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 1, 5. https://doi.org/10.1145/3295700

Software Effort Interval Prediction via Bayesian Inference and Synthetic Bootstrap Resampling

LIYAN SONG, Southern University of Science and Technology, China, and University of Birmingham, UK
LEANDRO L. MINKU*, University of Birmingham, UK
XIN YAO*, Southern University of Science and Technology, China, and University of Birmingham, UK

Software effort estimation (SEE) usually suffers from inherent uncertainty arising from predictive model limitations and data noise. Relying on point estimation only may ignore the uncertain factors and lead project managers (PMs) to wrong decision-making. Prediction intervals (PIs) with confidence levels (CLs) present a more reasonable representation of reality, potentially helping PMs to make better informed decisions and enable more flexibility in these decisions. However, existing methods for PIs either have strong limitations, or are unable to provide informative PIs. To develop a 'better' effort predictor, we propose a novel PI estimator called Synthetic Bootstrap ensemble of Relevance Vector Machines (SynB-RVM) that adopts Bootstrap resampling to produce multiple RVM models based on modified training bags whose replicated data projects are replaced by their synthetic counterparts. We then provide three ways to ensemble those RVM models into a final probabilistic effort predictor, from which PIs with CLs can be generated. When used as a point estimator, SynB-RVM can either significantly outperform or have similar performance compared with other investigated methods. When used as an uncertain predictor, SynB-RVM can achieve significantly narrower PIs compared to its base learner RVM. Its hit rates and relative widths are no worse than the other compared methods that can provide uncertain estimation.

CCS Concepts: •**Computing methodologies** →**Supervised learning by regression;** *Bayesian network models;* Ensemble methods; •**Software and its engineering** →**Software creation and management;**

Additional Key Words and Phrases: software effort estimation, software risk management, uncertain effort estimation, prediction intervals with confidence levels, Bootstrap resampling, relevance vector machine, synthetic replacement, Bootstrap resampling, ensemble learning

Reference format:
Liyan Song, Leandro L. Minku*, and Xin Yao*. 0000. Software Effort Interval Prediction via Bayesian Inference and Synthetic Bootstrap Resampling. 1, 1, Article 1 (0000), 43 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn.

1 INTRODUCTION

Software effort estimation (SEE) is the process of predicting the effort (e.g. in person-month or person-hour) required to develop a software system. It often takes place in the very early stage of software development, and is an important task in software project management. Based on it, project managers (PMs) can make key decisions for subsequent planning, control, bidding, and decision-making [8].

Uncertainty is inherent to SEE [35, 46]. There are several sources of uncertainty in the context of SEE. For instance, when using machine learning for creating SEE models, uncertainty may arise from the effort estimation model limitations and from the noise (e.g. data collection mistakes) in

Authors' emails: Liyan Song: songly@sustc.edu.cn or lxs189@cs.bham.ac.uk; Leandro L. Minku: L.L.Minku@cs.bham.ac.uk; Xin Yao: xiny@sustc.edu.cn or x.yao@cs.bham.ac.uk. *Corresponding authors.

0000. XXXX-XXXX/0000/0-ART1 $15.00
DOI: 10.1145/nnnnnnn.nnnnnnn.

the data set used for creating the SEE models [36, 41, 43, 50]. Uncertainty leads to many difficulties in SEE, especially in early project development phases [49]. Therefore, appropriate SEE methods should ideally handle uncertainty explicitly and support decision makers in assessing corresponding project risks based on such uncertainty. For decades, software estimation experts have pointed out that besides point estimates, effective project management also requires information about effort-related project risk [49]. In particular, they have suggested that an effort estimation for a predicted software project should be a range of values (e.g. prediction interval) with a specific probability (e.g. confidence level), within which the software development can be completed, rather than a single point estimation [43].

There is a vast literature on using machine learning for creating SEE models [8, 86]. However, most existing SEE methods produce only point estimation, i.e., they provide a single prediction of the effort required to develop a given project. The problem of proving only point estimate is that there is no feasible way to manage risks and uncertainty based on the point effort estimate. If a point prediction had to ensure against all possible risks and uncertainty, the price of constructing such a model would be prohibitive [43]. Therefore, relying on a point estimation may ignore uncertain factors and lead project managers to wrong decision-making. Besides a point estimate that is the most likely effort for a project, SEE methods should ideally support the handling of estimation uncertainty by accessing the probability of falling within a specific interval consisted of a minimum and a maximum effort values (i.e. a *prediction interval*). The certainty on this prediction interval can be characterized by a *confidence level* (CL). This can be considered as a more reasonable representation of reality than mere point estimates, thus helping project managers to make better informed decisions. It can also embrace the fact that effort estimates are probabilistic assessments of a future condition [49].

Prediction interval (PI) can be considered as a representative form of uncertain estimation, which allows for risk management and provide more flexibility to PMs. For instance, when bidding for a project, if the competition is very fierce the project manager can report a lower price within the interval to enhance the winning chances; when the competition is less fierce, he/she can propose a higher price for bringing more profit to the organization. In summary, it would be good to have PIs associated with CLs in the context of SEE [28, 40].

Recently, a probabilistic model called Relevance Vector Machine (RVM) [81] was introduced in the context of SEE, and provided a simple way of constructing PIs with CLs [79]. When used as a point estimator, RVM was shown to be very competitive compared with state-of-the-art effort methods. However, when used as uncertain predictor, the derived PIs were sometimes too wide to be informative. In this paper, we propose a novel SEE model based on RVM that can provide better PIs: on the one hand, the derived PIs should be wide enough to capture the actual efforts of many projects to be predicted; on the other hand, they should be sufficiently narrow to be informative and of practical use.

Inspired by ensembles that turn weak models into a stronger one in machine learning [1, 90], we adopt the Bootstrapping resampling technique to generate multiple 'weak' probabilistic RVM effort predictors, each of which is trained on part of the training projects. Due to sampling with replacement of the Bootstrap technique, each generated training bag contains replicated data projects, which makes RVM unable to compute the inverse of its kernel matrix needed by its training procedure. To address this problem, we propose a method to replace the repeated projects in each Bootstrap training bag with their synthetic counterparts, which are generated based on the original replicated project and its neighbours. We then propose three novel ways of combining the weak probabilistic effort models into a single one in order to obtain a 'stronger' and 'better' final predictor. We name our method as *synthetic Bootstrap ensemble of RVMs* (SynB-RVM).

The main contribution of this paper is to propose and validate SynB-RVM in terms of point and uncertain estimation. We also present a comprehensive overview of state-of-the-art methods that can provide PIs, and a thorough comparison among SynB-RVM and these methods. To the best of our knowledge, this is the most thorough experimental comparison on this topic. In summary, this paper investigates the following research questions:

- RQ1 When used as a point estimator, how well can SynB-RVM perform in comparison with other point and uncertain effort prediction methods that have been shown to perform well? This comparison enables us to check how promising SynB-RVM's (and other uncertain prediction methods') effort prediction is in comparison to state-of-the-art and baseline point effort estimation methods.
- RQ2 When used as an uncertain estimator, does SynB-RVM provide reasonable PIs? This RQ has two parts: (1) do the proposed PIs adequately cover the actual efforts of the testing projects? And (2) are the proposed PIs sufficiently narrow so that they can be informative and of practical use? This is our main objective and allows us to evaluate how well our goal of developing a 'better' uncertain predictor has been achieved.
- RQ3 If SynB-RVM can improve the point and uncertain estimation of its base learner RVM, which of its components contribute to the improvement? This allows us to gain a better understanding of SynB-RVM and find the reasons why it outperformed its base learner, contributing to the external validity of this study.

Our experimental studies based on the data sets from the SEACRAFT (former PROMISE) [58] and ISBSG [31] repositories show supportive performance of SynB-RVM. It achieves competitive or even superior point estimates in comparison to other methods, and produces better PIs over the uncertain methods investigated. Our analyses show the benefits of synthetic displacement and Bootstrap pruning components. By using our *synthetic Bootstrap ensemble* model, the software manager can obtain not only the most likely effort value but also a predictive range for the testing projects automatically, enabling flexibility in the bidding process and risk management.

The remainder of this paper is organized as follows. Section 2 briefly introduces background knowledge required in this work. Section 3 provides a comprehensive review of previous work that provide uncertain effort estimation, including a discussion of their strengths and weakness. The proposed method is discussed in Sec. 4 in terms of its training phase and in Sec. 5 in terms of its prediction phase. The data sets used in the experiments are described in Sec. 6, followed by the experimental design including performance metrics, benchmark methods, and their parameter settings in Sec. 7. The evaluations and analyses of SynB-RVM are discussed in Sec. 8. Section 9 studies the effectiveness of the three components of SynB-RVM, providing a more thorough understanding. Section 10 discusses its implications to practice. Section 11 further discusses SynB-RVM. Section 12 discusses the threats to validity. The paper is concluded in Sec. 13.

2 BACKGROUND

2.1 Bootstrap Resampling and Bagging

Bootstrap is a resampling technique that can be used to estimate population statistics such as mean, median, and variance [24, 25]. It samples uniformly with replacement n observations from data set $\mathcal{D}$ of size n, generating a set of Bootstrap bags of size n. In other words, the generated data sets have the same data size as the original one.

By *sampling with replacement*, we mean the procedure where a random number generator independently selects integers $j_1, \cdots, j_n$ each of which equals any value between 1 and n with probability $1/n$. These integers determine which members of $\mathcal{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n | \mathbf{x}_j \in \mathbb{R}^d\}$ are selected to be in the new random samples. An obvious consequence of this sampling method is that every

sampled Bootstrap bag is likely to contain data points that appear more than once while others do not appear at all. In this paper, we will use Bootstrap resampling to generate Bootstrap training bags, based on which RVM models will be constructed.

Bootstrap **agg**regat**ing** (*Bagging*) is an ensemble approach designed to improve the stability and accuracy of machine learning models. It can reduce the prediction error when its base models are unstable [13]. Bagging generates multiple base models trained on different Bootstrap samples as follows. Given a training set $\mathcal{D}$ of size n, bagging generates M new training sets $\{\mathcal{D}^{(m)}|m = 1, \cdots, M\}$, each of size n, by sampling from $\mathcal{D}$ uniformly and *with replacement*. On each new training set $\mathcal{D}^{(m)}$, one base model is trained and thus we have M trained models via Bagging. The final prediction is the average of the base models' predictions. In this paper, we compare the proposed method with the Bagging ensembles in terms of point estimation.

2.2 Uncertain Prediction in SEE

In this paper, uncertainty is considered in probabilistic terms and originates from the unpredictable and non-deterministic nature of the future software projects. In particular, it is interpreted as the factors influencing software development effort. The uncertainty of effort estimations can be characterized through two types of interval predictions as shown below.

An effort *prediction interval* (PI) comprises a minimum and maximum values between which the future effort is expected to lie at a *confidence level* (CL). It is usually associated to a most likely point estimate. For instance, a project manager may be 95% certain that the predicted effort of a project will fall between 500 and 2,500 person-hour with the most likely effort value at 1,500.

Confidence interval (CI) is another uncertainty concept, which usually refers to the uncertainty associated with the unknown population statistics, such as the uncertainty of the mean value of an unknown distribution [5, pp.761-824]. For instance, a project manager may be 95% certain that the mean effort of all developed software projects is 1,500 person-month.

In summary, PIs are related with an unknown project to be predicted, while CIs are connected with the mean effort of existing projects. In this paper, we are more interested in providing PIs with CLs for an unknown project.

2.3 Relevance Vector Machine (RVM)

We briefly introduce the Relevance Vector Machine (RVM) [10, 26, 81]. For better understanding, we deliberately omit many details. Please refer to Chapter 7.2 of [10] and [81] for detailed deductions.

Relevance Vector Machine (RVM) [10, 26, 81] is a typical generalised linear model, and can be represented for output y given input vector $\mathbf{x}$ as

$$y = \boldsymbol{\theta}^T \boldsymbol{\Phi}(\mathbf{x}) + \epsilon, \tag{1}$$

where $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_N]$ are the model parameters to be learned during training process, ϵ denotes the uncertain information from actual effort collection and is assumed to be Gaussian distributed as $\epsilon \sim \mathcal{N}(0, \sigma^2)$, and $\boldsymbol{\Phi}$ is known as the *kernel matrix* that consists of *basis functions* on training samples and gives linear regression substantial flexibility for modelling nonlinear relation between $\mathbf{x}$ and y. The *basis function* measures the distance between a training sample and the project to be predicted. There are several choices for the basis functions. In this work, we adopt the non-normalized Gaussian basis function:

$$\phi_j(\mathbf{x}) = exp\{-(\mathbf{x} - \boldsymbol{x_j})^2/(2c^2)\} \tag{2}$$

where the $\boldsymbol{x_j}$ is the j-th training sample, and parameter c controls the *width* that can be determined using cross-validation method [10]. We choose this basis function due to the locality feature of SEE data [60]. It is noteworthy that RVM's training procedure involves the calculation of the inverse of

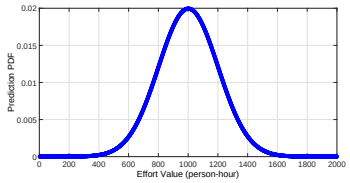

Fig. 1. An illustration of probabilistic prediction from RVM for a software project. The probabilistic prediction is Gaussian distributed with its most likely value at 1,000 person-hour.

the kernel matrix $\mathbf{\Phi}$, requiring a training set composed of different projects in order to avoid the invertibility problem.

Following the Bayesian framework, RVM first introduces a zero-mean Gaussian *prior* $p(\boldsymbol{\theta})$ to the model parameters $\boldsymbol{\theta}$. The prior is a belief on the SEE model before any observation and evidence is taken into account. The detailed shape of prior is governed by a set of hyper-parameters, one associated with each θ_n for $n \in \{1, \cdots, N\}$, with their most probable values iteratively estimated from the data. After this iterative procedure finishes, the training vectors corresponding to non-zero model parameters are called *relevance vectors* in line with the *support vectors* of Support Vector Machine (SVM), which is a popular learning method mainly designed for point estimation [82–84]. RVM can be considered as a Bayesian approach to SVM, which can provide a probabilistic predictions instead of a point estimate for a testing project.

Then, we can obtain the *posterior* of model parameters $p(\boldsymbol{\theta}|\mathcal{D})$, being a conditional probability after the training examples $\mathcal{D}$ are taken into account. The posterior of the model parameters is a Gaussian distribution proportional to the product of the Gaussian *prior* $p(\boldsymbol{\theta})$ and the Gaussian *likelihood* of all training samples $p(\mathcal{D}|\boldsymbol{\theta})$, which is calculated according to Bayes' Rule:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})}{p(\mathcal{D})}$$

Finally, we can have a *probabilistic* (Gaussian) effort prediction for a new project as shown in Fig. 1. A point estimation can be easily obtained by being assigned to the mean of the Gaussian distribution (e.g. 1,000 person-hour in Fig. 1), since it is the most likely effort value.

2.4 Automatically Transformed Linear Model (ATLM)

Automatically Transformed Linear Model (ATLM) is a simple linear model [87]. It works by developing a simple linear relationship between input features and output effort after appropriate automatic transformations upon them. Least square estimation [64] is used to estimate model parameters automatically. ATLM is a suitable baseline model for comparison against SEE predictors [87], which has been shown to be comparable or even superior to other more advanced SEE methods such as Pareto ensembles of ANNs [61] and the hybrid ABE-PSO [42]. We use the R codes provided by the authors for its implementation.

It is noteworthy that naively applying Multivariate Linear Regression (MLR) may not be adequate since SEE data is often skewed. Therefore, appropriate transformations of the inputs and/or output are often required to form a proper MLR [44]. To this end, ATLM assesses the suitability of logarithm and square-root transformations of each effort variable (inputs and output) based on the underlying distribution discovered from the training data. For each effort variable, the transformation (logarithm, square-root and none) that results in the least skewed data is applied to construct the final linear model. Skewness is measured by the $b1$ metric proposed in [32]. ATLM can automatically decide when to apply what transformation and is thus superior to MLR. In particular, MLR is a special case of ATLM when no transformations are applied.

3 RELATED WORK

Few studies have considered the development of automated models that are able to provide uncertain predictions for SEE. They can be cast into the following five categories.

3.1 Bootstrap Wrapping

Angelis et al.'s [2] was the first attempt to suggest uncertain effort prediction, where the authors compared the effort predictions derived from a Bootstrap-based model with the ones from regression-based methods. However, the method was actually producing CIs (of the mean effort of existing projects) rather than the PIs (of new observations) [3, 34]. Later, Bootstrap resampling was integrated with a hybrid software model called CoBRA [14] in order to provide PIs for SEE [50]. The authors wrapped Bootstrap resampling into the CoBRA training process by replacing a single *nominal project* of CoBRA with an empirical distribution. To construct CoBRA, domain experts were asked to decide the *causal factors* of CoBRA and their possible values. Experimental results showed realistic uncertainty estimates. However, this method requires intensive human participation in constructing CoBRA and is very specific to CoBRA. More recently, Laqrichi et al. [52] considered uncertainty when using neural network (NN) for SEE via Bootstrap mechanism. The proposed method generated a probability distribution of point estimates, based on which the PIs can be computed. The empirical results showed better point estimation compared to traditional effort estimation based on linear regression.

The uncertain effort predictors in this category wrap Bootstrap resampling to reproduce multiple training sets [52] or estimates of model components [50], from which the effort PIs can be computed. They are different from our method in the following ways. (1) *Base learner:* Usually, their base learners only provide point estimates [2, 52]. In contrast, our method is based on probabilistic predictors, from which the uncertainty can be retained and calculated automatically leading to ideally more sensible uncertain predictions. (2) *Usage of Bootstrap bags:* They use all generated bags regardless of their resulting predictions being unreasonable, whereas our method prunes those unreasonable ones.

3.2 Empirical Error Probability Consistency Assumption

Jørgensen and Sjøberg [40] proposed and evaluated a simple effort PI method, based on the assumption that empirical distribution of estimation accuracy was consistent between the historical and the predicted projects. Comparisons between the proposed method, a regression-based and human judgement-based effort PIs showed that different methods could perform well in some data while failed in others. Another work [12] with the same assumption was proposed later, but aims to produce CIs rather than PIs.

The uncertain effort predictors in this category assume that the estimation accuracy of earlier software projects predicts the uncertainty of new projects. However, when this is not the case the results are misleading. In contrast, the source of uncertainty of our method is assumed to originate from the Gaussian noise assumption on the observed efforts, which lays its foundation on *central limit theorem (CLT)* [65], stating that the summation of several independent random processes tends to a normal distribution even if the original variables themselves are not normally distributed. Considering the errors/noises that generate model uncertainty as random variables, their overall effect is reasonable to be simulated by Gaussian distribution. However, this assumption still has problems for disregarding the fact that effort values have to be positive. Better performance can be expected with a more proper noise assumption. More discussions are provided in section 11.

3.3 Categorical Conversion

Sentas et al. [73, 74] employed ordinal regression to classify a new project into a predefined effort category (e.g. low, nominal or high). The historical completed projects were required to predefine the spent effort categories. The point estimation was the mean or median value of the category the predicted project falls in. Empirical comparisons between the models producing point and predefined interval estimates were conducted in [9], but found no general conclusions as the best performed method could behave relatively bad depending on the data set. Later in [7], clustering analysis was applied to automatically define the effort categories. Its main contribution was the removal of human intervention in predefining the effort intervals.

The uncertain effort predictors in this category have the following problems: (1) The performance of the uncertain prediction heavily relies on the goodness of the predefined intervals. (2) They suffer inferior point performance as they simply use median or mean of the categorical intervals for the corresponding point estimates. (3) The interval predictions and point estimates would be exactly the same for all projects in the same category, which may be improper and highly limit the representativeness of effort predictions. (4) It is hard to interpret the category intervals and the confidence (i.e. *confidence level*) of these intervals is not provided.

Recently, Mensah et al. proposed a more advanced method of this category that automatically determined the categorical intervals [56]. The method provides duplex outputs for a testing project: one for the *effort estimate* as typically done in SEE studies and one for the effort level (high, moderate or low) for interpretation purpose. Their method discretises the training efforts into high, moderate, or low levels according to the density quantile function, and the estimated effort of a testing project is subsequently assigned to one of these defined levels. This method allows the researchers and practitioner for better interpretation of the prediction results.

Both SynB-RVM and Mensah et al.'s method [56] can provide additional information of the testing project helping the PMs for better decision making. However, they mainly differ in the catering problems and the types of information provided. Mensah et al.'s method aims to improve the interpretability of effort point estimates and thus provides duplex outputs - one for the point estimation and the other for its high/moderate/low level; in contrast, our method aims to cater the inherent uncertainty within the SEE data and to support the PMs in their decision making by providing interval predictions together with effort point estimates.

3.4 Uncertain Prediction from Bayesian Inference

Recently, a Bayesian regression model, namely RVM, was introduced to SEE [79], which provided a simple way of constructing PIs with CLs [79]. Specifically, based on the properties of the Gaussian distribution derived by RVM, the PI with any CL $\alpha \in (0, 1)$ was presented based on the cumulative distribution function (CDF) of the derived Gaussian effort estimation. Empirical results showed very competitive performance compared to state-of-the-art SEE methods in terms of point prediction performance. However, the derived PIs were sometimes too wide to be informative [79].

There are some other methods based on Bayes' Theorems [18, 55, 68, 80] to infer software uncertain prediction. However, these methods do not aim to and cannot provide effort PIs, and is thus out of the scope of this paper. In this paper, we aim to improve the performance of RVM-based PI methods in terms of a narrower and more informative PI and at the same time maintain or improve the point estimate performance.

3.5 Other Methods Optimizing Uncertainty

Sarro et. al. [71] proposed a bi-objective effort estimator to optimise the accuracy of the point estimates and the uncertainty associated with the estimation model simultaneously. Their method

ALGORITHM 1: Synthetic Project Displacement

1: **Input:** (1) A software project $(\mathbf{x}, y)$ that has been resampled K times in $\mathcal{D}^{(m)}$, and (2) the initial training pool $\mathcal{D}' = \mathcal{D}$.
2: **Aim:** Retain one copy of $(\mathbf{x}, y)$ and replace its K-1 repetitions with the synthetic counterparts.
3: **Procedures:**
4: (1) Find the *furthest neighbour* $\mathbf{x}'$ of $\mathbf{x}$ in $\mathcal{D}'$ by Euclidean distance. To avoid scalability problem, each feature is standardized to have zero-mean and unit-variance.
5: (2) The synthetic project is generated as a linear combination of $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ as:

$$\begin{cases} \mathbf{x}^{syn} = (1-\rho)\mathbf{x} + \rho\mathbf{x}', \\ y^{syn} = (1-\rho)y + \rho y', \end{cases} \tag{3}$$

where the parameter $\rho \in (0, 1)$ controls the displacement degree.
6: (3) Replace one copy of $(\mathbf{x}, y)$ with $(\mathbf{x}^{syn}, y^{syn})$ in $\mathcal{D}^{(m)}$. Reset $\mathcal{D}'$ to be $\{\mathcal{D}' - (\mathbf{x}', y')\}$.
7: (4) Repeat step (1)~(3) until all (K-1) repetitions are replaced by their synthetic counterparts.
8: **Output:** The revised Bootstrap bag $\mathcal{D}^{(m)}$ with non-repeatable project $(\mathbf{x}, y)$.

aimed to build a robust model by decreasing the uncertainty while the model evolution. However, it cannot provide uncertain predictions, and is thus out of the scope of this paper.

4 TRAINING PHASE OF SYNB-RVM

Consider a training set of N software projects $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ where the feature vector $\mathbf{x}_n \in \mathbb{R}^d$ represents the software features such as software development type, programming language, and team expertise, and y_n is the actual effort for developing this software. In the training phase, several RVMs are trained with the following three steps.

4.1 Bootstrap Training Bag Construction

By using Bootstrap resampling with replacement on the original data set $\mathcal{D}$, SynB-RVM creates M Bootstrap training bags of size N, denoted as $\{\mathcal{D}^{(m)}\}_{m=1}^{M}$. *Sampling with replacement* is reasonable for SEE because it is a small data problem, and thus resampling will not take an excessive amount of time. Each Bootstrap bag $\mathcal{D}^{(m)}$ will be used to develop one RVM estimator.

4.2 Synthetic Project Displacement

Every $\mathcal{D}^{(m)}$ is likely to contain duplicated data due to sampling with replacement to create bags with size N. The replicated projects will cause invertibility problem of the kernel matrix when training RVMs [81]. To this end, we propose a displacement technique to generate synthetic projects to replace these repetitions. The effectiveness of the synthetic replacement technique in improving the point and uncertain prediction performance is verified in section 9.2.

Suppose that a training project $(\mathbf{x}, y) \in \mathcal{D}^{(m)}$ has been resampled K times. We retain one copy of it and displace all the others along certain directions to form (K-1) different *synthetic software projects* as shown in Algorithm 1. It can be interpreted as a shift of the replicated project towards a different but similar *data cloud* in the SEE training space. After replacing all repeated training projects, we obtain a non-repeatable revised Bootstrap bag $\mathcal{D}^{(m)}$.

The reasons for displacing the repeated project along its furthest neighbour in SEE data are twofold: (1) Choosing the furthest neighbour suggests a more *diverse* Bootstrap training bag and thus is more likely to relieve the invertibility problem. It may also enhance the capability of the effort estimator for heterogeneous SEE data. (2) Disturbance of the repeated project by using another real SEE data can avoid the synthetic project to be too far away from the actual SEE data.

ALGORITHM 2: Training Phase of SynB-RVM

1: **Aim:** Train M RVMs that are used in prediction phase.
2: **Input:** (1) Training software projects $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, (2) the number of Bootstrap bags M, and (3) the degree of synthetic displacement ρ.
3: **Procedures**
4: (1) Bootstrap Training Bag Construction: Create M Bootstrap training bags from the data set $\mathcal{D}$ by using Bootstrap resampling with replacement.
5: (2) Synthetic Project Generation: For each Bootstrap training bag, replace the repeated training projects with their synthetic counterparts as Algorithm 1.
6: (3) RVM Training: Train M RVM models, each of which is based on one revised Bootstrap training bag. The M RVM models can be trained in parallel.
7: (4) Calculate the training errors of the M RVM models according to some performance metric.
8: **Output:** (1) M trained RVM models and (2) their training errors.

It is noteworthy that synthetic projects we generate may not and are not necessary to be composed of 'real' software features. For instance, some synthetic feature may be decimal for an ordinal feature due to a linear combination of two integers. To keep the notation simple, we use $\{\mathcal{D}^{(m)}\}$ to denote the revised Bootstrap bags from this point onwards.

4.3 RVM Training

The last step of the training phase is to develop RVM estimators from Bootstrap training bags $\{\mathcal{D}^{(m)}\}$. One RVM is trained on each Bootstrap training bag $\mathcal{D}^{(m)}$ using the training procedure described in Sec. 2.3. The M RVM models can be trained in parallel. The training phase of SynB-RVM is summarised in Algorithm 2.

5 PREDICTION PHASE OF SYNB-RVM

For a testing software project $\mathbf{x}$ with unknown effort y, the prediction phase aims to provide PIs with CLs that are wide enough to capture its actual effort and at the same time sufficiently narrow to be informative of practical use based on the trained RVMs. A second aim of prediction phase is to provide competitive point estimate in comparison with RVM-related methods and SEE methods that have been shown to perform well. Our prediction phase consists of the following four steps.

5.1 Bootstrap Uncertain Estimates

From the trained RVM models, we can obtain M Gaussian PDFs $\{\mathcal{N}(y^{(m)}, \sigma^{(m)})\}$ as the probabilistic estimates for the testing project $\mathbf{x}$, where each $y^{(m)}$ and $\sigma^{(m)}$ are the Gaussian mean and standard deviation (std) respectively for Bootstrap bag $\mathcal{D}^{(m)}$. To generate the final probabilistic software prediction, we will combine these PDFs. As Gaussian distribution is uniquely determined by its mean and std, this issue can be simplified into combining M pairs of $\{(y^{(m)}, \sigma^{(m)}\}$.

5.2 Bootstrap Estimate Pruning

Before framing the final prediction, we note that: (1) some $\{y^{(m)}\}$ may be improperly negative due to the base RVMs being weak learners, and (2) the estimates from some Bootstrap bags may not perform well in the training set and are improper to be retained in the prediction phase. Thus, it would be reasonable for our proposed method to prune these improper Bootstrap bags before constructing the final estimate.

ALGORITHM 3: Prediction Phase of SynB-RVM

Aim: Provide the PI with any CL for a testing project.
Input: (1) CL α, (2) the project being estimated $\mathbf{x}$, (3) the trained RVM models, (4) the training errors of these RVM models and (5) the pruning rate τ.
Procedures
    (1) Bootstrap Uncertain Estimates: Compute the M Gaussian PDFs for the testing project $\mathbf{x}$ using the M trained RVMs and denote them as $\{\mathcal{N}(y^{(m)}, \sigma^{(m)})\}_{m=1}^{M}$.
    (2) Bootstrap Estimate Pruning: Prune those RVMs with (a) negative estimated mean values, and (b) bad training performance in term of mean absolute error.
    (3) Final Probabilistic Prediction: Three methods to calculate the final probabilistic prediction for the testing project $\mathbf{x}$ using Eq. (4)~(6).
    (4) PI Construction: Convert the derived Gaussian PDF prediction to CDF and derive the PI $[y_{lb}, y_{ub}]$ with CL α using Eq. (7) or Eq. (8).
Output: PI $[y_{lb}, y_{ub}]$ with CL α.

5.2.1 Pruning RVM Training Bags with Negative Estimated Mean. According to background knowledge, software effort values should be positive, and thus those bags with negative point estimates will be pruned.

5.2.2 Pruning RVM Training Bags with Bad Training Performance. According to machine learning theory, high training error usually indicates bad prediction performance [10], and thus those bags with high training errors will be pruned. We rank Bootstrap bags according to their performance of point estimates on training data, and then prune those bags that are the worst $\tau \in [0, 1]$ percentage. People can choose the pruning performance metric based on practical preference. In our implementation, we use mean absolute error in line with our main evaluation metric of point prediction for its being unbiased towards under/over-estimation. Denote $M' \leq M$ as the number of remaining Bootstrap bags.

5.3 Final Probabilistic Prediction

We propose three methods for deriving the final probabilistic prediction based on $\{(y^{(m)}, \sigma^{(m)})\}_{m=1}^{M'}$.

5.3.1 Empirical Mean. One of the simplest ways to derive the final probabilistic estimate is the sample means of these Bootstrap estimates:

$$\begin{cases} \hat{y} = \frac{1}{M'} \sum_{m=1}^{M'} y^{(m)}, \\ \hat{\sigma} = \frac{1}{M'} \sum_{m=1}^{M'} \sigma^{(m)}. \end{cases} \tag{4}$$

5.3.2 Uni-variant Empirical PDFs. We simulate the PDF of $\{y^{(m)}\}$ and $\{\sigma^{(m)}\}$ based on the estimations provided by the trained RVM models. Then, we set the mean and std of the final probabilistic estimate as the expectations of those two PDFs respectively. In our setting, we first develop the frequency histograms for $\{y^{(m)}\}$ ($\{\sigma^{(m)}\}$), where the number of bins B is automatically determined by the binning algorithm[1] with uniform width that can cover the range of elements and reveal the underlying shape of the distribution. Then, we characterize the b-th bin by its middle point $y(b)$ ($\sigma(b)$) and calculate its frequency $f_y(b)$ ($f_\sigma(b)$). Finally, the mean and std of the final probabilistic prediction are calculated as:

$$\begin{cases} \hat{y} = E\{y(b)\} = \sum_{b=1}^{B} y(b) \cdot f_y(b), \\ \hat{\sigma} = E\{\sigma(b)\} = \sum_{b=1}^{B} \sigma(b) \cdot f_\sigma(b). \end{cases} \tag{5}$$

[1] See Matlab's histogram() function.

Table 1. The Investigated Data Sets

Repository	Name	#(Project)	#(feature)
SEACRAFT	Maxwell	62	23
	Kitchenham	145	3
	Cocomo81	63	17
	Nasa93	93	17
ISBSG	Org1	76	3
	Org2	32	3
	Org3	162	3
	Org4	122	3
	Org5	21	3
	Org6	22	3
	Org7	21	3

5.3.3 Bi-variant Empirical PDFs. Similar to the method described in Sec. 5.3.2, this method is also based on empirical PDFs. However, bi-variant empirical PDF is used. In this way, the correlation between y and σ can be taken into account. First, develop the 2D frequency histogram for $(\{(y^{(m)}, \sigma^{(m)})\})$, denoted by $f_{(y,\sigma)}(b_1, b_2)$, for which the numbers of bins (B_1, B_2) are automatically determined by the binning algorithm[2] to cover the data range and reveal the shape of the underlying distribution. Then, characterize each rectangle bin by its geometric middle point $\{(y(b_1), \sigma(b_2))\}$ and calculate its frequency $f_{(y,\sigma)}(b_1, b_2)$. Finally, mean and std of the final probabilistic prediction are calculated as:

$$(\hat{y}, \hat{\sigma}) = E\{(y(b_1), \sigma(b_2))\} = \sum_{b_1, b_2} (y(b_1), \sigma(b_2)) \cdot f_{(y,\sigma)}(b_1, b_2). \tag{6}$$

5.4 Prediction Interval Construction

Denote $\widehat{y}$ as the final predicted mean and $\widehat{\sigma}$ as the final predicted std from one of Eq. (4)~(6). Since the final predictive estimation is Gaussian $\mathcal{N}(\hat{y}, \hat{\sigma})$, the PI with any CL α can be calculated as:

$$PI_\alpha = [max\{0, \mathcal{F}^{-1}(\frac{1-\alpha}{2})\}, \mathcal{F}^{-1}(\frac{1+\alpha}{2})], \tag{7}$$

where $\mathcal{F}^{-1}(\beta)$ represents the effort value located on the β percentile of this Gaussian *cumulative distribution function* (CDF). In particular, based on "68-95-99.7" rule of Gaussian distribution [89], the PIs with $CL_{0.6827}$, $CL_{0.9545}$, and $CL_{0.9973}$ can be simply derived as:

$$[max(0, \widehat{y} - j\widehat{\sigma}), \widehat{y} + j\widehat{\sigma}], \tag{8}$$

for $j \in \{1, 2, 3\}$ respectively. It is worth noting that we could also derive PIs with $CL_{0.6827}$, $CL_{0.9545}$, and $CL_{0.9973}$ according to Eq. (7), and it is just easier when derived by Eq. (8). The testing phase of SynB-RVM is summarized in Algorithm 3.

6 DATA SETS

The analyses of this paper are based on data sets from the Software Engineering Artifacts Can Really Assist Future Tasks (SEACRAFT) Repository [58][3] and the International Software Benchmarking Standards Group (ISBSG) Repository [31] Release 10, which are chosen to cover a wide range of features, such as number of projects and type of features, countries, and companies. Table 1

[2]See Matlab's histcounts2() function.

[3]The data we are using were previously in the PROMISE repository [57].

contains a basic description of these data sets. Sections 6.1 and 6.2 provide detailed description and explanation on their preprocessing procedures.

6.1 SEACRAFT Data

6.1.1 Maxwell [21]. This data set was first presented in [54] for illustration of linear regression models in SEE, and then described in [74] for ordinal regression models. Maxwell contains 62 projects from one of the biggest commercial banks in Finland, covering the years from 1985 to 1993 and both in-house and outsourced development. The following steps were performed to process this data set for use in this work:

(1) *Features:* Remove the input features start year (*syear*) and *duration* (*duration* = *syear* − 1985 + 1). Start year was removed following the same preprocessing as [74] since it was found to have no significant effect on the dependent effort according to one-way ANOVA. Duration was removed because we usually could not know the project delivery time in reality during effort prediction process. This preprocessing resulted in the 23 input features listed in table 2.
(2) *Categorical conversion:* The categorical features are converted into numerical values so that all the investigated methods are applicable on the same data set.
(3) *Normalization:* Normalize each of the 23 input features to have zero-mean and unit-variance. Zero-mean preprocessing can usually simplify ML methods and unit-variance of each feature can avoid scalability problem among different features.
(4) *Missing values:* There were no missing values in this data set.
(5) *Output:* The output effort was measured in hours and remained unchanged.

6.1.2 Kitchenham [20]. The detailed description for Kitchenham data set can be found in [45]. It comprises 145 maintenance and development projects undertaken between 1994 and 1998 by a single software development company. The following steps were performed to process this data set for use in this work:

(1) *Features:* Remove the input features project ID, actual start date, actual duration, estimated completion date, first estimate, and first estimate method. Project ID was removed because it was irrelevant for training a SEE model. Actual start date was removed following the same preprocessing as [45]. Completion date together with start date would give the duration of the project, and duration was removed because it was considered as a dependent variable. The other features were removed because they were themselves estimations of completion date or effort, or represent the method used for such estimations. This preprocessing resulted in the three remaining input features: adjusted function points, project type and client code listed in table 3.
(2) *Categorical conversion*: The categorical features are converted into numerical values, for the same reason as for Maxwell.
(3) *Normalization:* Normalize each of the 3 input features to have zero-mean and unit-variance, for the same reason as for Maxwell.
(4) *Missing values:* Treat missing values using 1-NN imputation method that had shown to improve SEE [15]. This imputation method is based on k-nearest neighbours (k-NN). It first finds the k most similar complete projects to the target project to be imputed where similarity is measured by Euclidean distance. After that, the missing values for the feature are assigned with the same values of their nearest neighbours or determined by vote counting when $k > 1$. There were in total ten projects with missing values.
(5) *Output:* The output effort was measured in hours and remained unchanged.

Table 2. Detailed description on Maxwell features.

ID	name	description	type	value
1	App	application type	categorical	{information/online service, transaction service, customer service, management info system, production control logistics order processing}
2	Har	hardware platform	categorical	{mini computer, multi-platform, networked, mainframe, PC}
3	Dba	database	categorical	relational, sequential, other, none
4	Ifc	user interface	categorical	GUI, text user interface
5	Source	where developed	categorical	in-house, outsourced
6	Telonuse	Telon use	categorical	No, Yes
7	Nlan	#development languages	ordinal	{1 = one language used, 2 = two languages used, 3 = three languages used, 4 = four languages used}
8	T01	customer participation	ordinal	{1 = very low, 2 = low, 3 = normal, 4 = high, 5 = very high}
9	T02	development environment adequacy	ordinal	
10	T03	staff availability	ordinal	
11	T04	standards use	ordinal	
12	T05	methods use	ordinal	
13	T06	tools use	ordinal	
14	T07	software's logical complexity	ordinal	
15	T08	requirements volatility	ordinal	
16	T09	quality requirements	ordinal	
17	T10	efficiency requirements	ordinal	
18	T11	installation requirements	ordinal	
19	T12	staff analysis skills	ordinal	
20	T13	staff application knowledge	ordinal	
21	T14	staff tool skills	ordinal	
22	T15	staff team skills	ordinal	
23	size	application size	numerical	continuous

Table 3. Detailed description on Kitchenham features.

ID	name	description	type	value
1	func	adjusted functional size	numerical	continuous
2	proj	project type	categorical	A, C, D, P, Pr, U
3	client	client code	categorical	C1, C2, C3, C4, C5, C6

6.1.3 Cocomo81 and Nasa93 [22]. The two data sets follow the COCOMO [11] data format, which has 17 input features consisting of 15 cost drivers, lines of code (*loc*) and the development type. The detailed description can be found in table 4. The data sets were processed to use the COCOMO numeric values for the cost drivers. Cocomo81 consists of 63 projects analysed by Boehm to introduce COCOMO model [11]. Nasa93 contains 93 Nasa projects developed between 1970's and 1980'. The following steps were performed to process this data set for use in this work:

(1) *Categorical conversion:* The categorical features are converted into numerical values, for the same reason as for Maxwell.
(2) *Normalization:* Normalize each of the 17 features to have zero-mean and unit-variance, for the same reasons as for Maxwell.
(3) *Missing values:* There were no missing values in these two data sets.
(4) *Output:* The output effort was measured in person-month and remained unchanged.

Table 4. Detailed description on COCOMO-format features [11]. The term 'corr' denotes correlation to effort according to original description. In particular, 'U-shaped' correlation to effort means giving programmers either too much or too little time to develop a system can be detrimental.

ID	name	description	corr	type	value
1	loc	line of codes	none	numerical	continuous
2	dev	develop type	none	categorical	{organic, embedded; semidetached}
3	rely	required software reliability	pos	ordinal	{1 = very low; 2 = low; 3 = normal; 4 = high; 5 = very high; 6 = extra high}
4	data	data base size	pos	ordinal	
5	cplx	process complexity	pos	ordinal	
6	time	time constraint for CPU	pos	ordinal	
7	stor	main memory constraint	pos	ordinal	
8	virt	machine volatility	pos	ordinal	
9	turn	turnaround time	pos	ordinal	
10	acap	analysts capability	neg	ordinal	
11	aexp	application experience	neg	ordinal	
12	pcap	programmers capability	neg	ordinal	
13	vexp	virtual machine experience	neg	ordinal	
14	lexp	language experience	neg	ordinal	
15	modp	modern programming practices	neg	ordinal	
16	tool	use of software tools	neg	ordinal	
17	sced	schedule constraint	U-shape	ordinal	

6.2 ISBSG Data

ISBSG release 10 contains a large body of completed software projects (5,052 projects), covering many different companies, several countries, organisation types, application types, etc. The data can be used for different purposes, such as evaluating the benefits of changing a software or hardware development environment, improving practices and performance, and estimation [31].

First, we preprocessed the ISBSG repository following the same procedures as [60], resulting in 621 projects, by maintaining only projects with:

- Data and function points quality A (assessed as being sound with nothing being identified that might affect their integrity) or B (appears sound but there are some factors which could affect their integrity/integrity cannot be assured).
- Recorded effort that considers only development team.
- Normalize effort equal to total recorded effort, meaning that the reported effort is the actual effort across the whole life cycle.
- Functional sizing method IFPUG version 4+ or NESMA.
- No missing organisation type field.

After that, a set of relevant comparison data sets need to be selected in order to produce reasonable SEE using ISBSG data. The selected projects were grouped into several ISBSG data sets according to the *organisation type* [60], and only the groups with at least 20 projects were maintained, following ISBSG's data set size guidelines. The resulting organisation types are shown in table 5.

Finally, we performed the following steps to process these ISBSG data sets for use in this work:

(1) *Features:* The ISBSG suggests that the most important criteria for estimation purpose are the functional size, the development type (new development, enhancement or re-development), the primary programming language (3GL, 4GL or ApG) and the development platform (mainframe, midrange or PC). As development platform is missing in more than 40% of

Table 5. ISBSG data sets grouped according to organization type and only the groups with at least 20 projects were maintained following ISBSG's data size guideline.

ID	Organisation Type	#Projects
1	financial, property & business services	76
2	banking	32
3	communications	162
4	government	122
5	manufacturing, transport & storage	21
6	ordering	22
7	billing	21

Table 6. Detailed description on ISBSG features.

ID	name	description	type	value
1	func	functional size	numerical	continuous
2	dev	development type	categorical	{enhancement, re-development, new development}
3	lang	primary programming language	categorical	{3GL, 4GL, ApG}

the projects for two organisation types, the remaining three criteria were used as input features listed in table 6.

(2) *Categorical conversion:* The categorical features are converted into numerical values, for the same reason as for Maxwell.

(3) *Normalization:* Normalize *functional size* to have zero-mean and unit-variance, for the same reason as for Maxwell.

(4) *Missing values:* There were no missing values for the features of *functional size* and *development type*, but *language type* had missing values across several data sets. For instance, Org1 had 25 out of 76 projects (about 33%) with their values of *language type* missing. We would lose too many data that were potentially useful in improving and evaluating a model's performance if we further eliminated those projects [59]. Thus, instead of discarding the projects in which the values of *language type* were absent, we treated these missing values by 1-NN imputation method [15], with the same procedures as in Kitchenham.

(5) *Output:* The output effort was measured in hours and remained unchanged. Due to the preprocessing, this is the actual development effort across the whole life cycle.

7 EXPERIMENTAL DESIGN

The experiments are designed to answer RQ1-RQ3. In order to answer RQ1, the comparisons between SynB-RVM and the methods that can provide uncertain prediction together with the state-of-the art point estimators will be made in Sec. 8.1. This comparison is to show how promising SynB-RVM's estimations (and other uncertain prediction methods' estimations) are compared with existing methods used for point effort estimations. In order to answer RQ2, the comparison of the PIs produced by SynB-RVM and the other uncertain methods will be made in Sec. 8.2. This comparison is to investigate whether the proposed method can produce improved uncertain performance. In order to answer RQ3, the comparisons between SynB-RVM and its variants are made in Sec. 9. These comparisons and analyses are to explore the effectiveness the components of SynB-RVM in giving better point and uncertain performance.

In this section, we will describe our experimental design including the performance metrics, compared methods, and parameter settings investigated.

7.1 Performance Metrics

7.1.1 Metric for Point Prediction. There are several performance metrics that can be used for empirical evaluation of point SEE models. Popular examples are Mean Absolute Error (MAE), Median Absolute Error (MdAE), Mean Magnitude of the Relative Error (MMRE), Percentage of estimations within $N\%$ of the actual value (Pred(N)), Logarithmic Standard Deviation (LSD) [27], and Standardised Accuracy (SA) [76].

Different performance metrics emphasize different factors of predictions and can behave differently in evaluating effort models [61]. For instance, MMRE, which is based on the magnitude of the relative error, $\frac{1}{N}\sum_{i=1}^{N}|y_i-\widehat{y_i}|/y_i$, was shown to be biased towards prediction systems that underestimate effort and could be misleading [27, 47, 63, 76]. Underestimation (over-optimism) is the direction of the error that practitioners are more unwilling to see [37, 39]. In contrast, MAE is a symmetric metric, not bias towards under- or over-estimation [76]. In fact, people should choose the performance metric according to their particular emphasis and practical interests.

MAE has been recommended by Shepperd and MacDonell [76] for SEE studies, and is adopted for evaluating point estimates in this paper. It measures the average magnitude of the errors as:

$$\sum_{i=1}^{N}|y_i-\widehat{y_i}|/N, \tag{9}$$

where $y_i/\widehat{y_i}$ denotes the actual/estimated effort, and N is the number of testing projects. It is noteworthy that we need to employ MAE in the development of our prediction system in line with the same choice of performance metric (Sec. 5.2).

MdAE has shown to be less sensitive than MAE to occasional projects with very large efforts and is a useful addition to MAE [27]. It is defined as the median value of the prediction residues $\{|y_i-\hat{y}_i|, i=1,\cdots,n\}$.

LSD is chosen for being a more reliable criterion than MMRE or MBRE (mean BRE) [27] as being in the logarithm scale of effort values. It is defined as

$$LSD=\sqrt{\frac{\sum(r_i+\frac{s^2}{2})^2}{n-1}}.$$

where $r_i = ln\, y_i - ln\, \hat{y}_i$, and s^2 is the variance of these residuals $\{r_i\}$. Smaller LSD values correspond to better point prediction performance.

SA is chosen for providing interpretable results. Given n effort values $\{y_i\}_{i=1}^{n}$ and their estimates $\{\widehat{y_i}\}_{i=1}^{n}$ predicted by method P, SA is defined as [76]

$$SA=1-\frac{MAE_P}{\overline{MAE}_{P_0}},$$

where baseline method P_0 denotes the random guessing, and $\overline{MAE}_{P_0}$ is the prediction performance (measured in MAE) of a large number (typically 1000) runs of random guessing. Estimating $\widehat{y_t}$ by P_0 is to take $\widehat{y_t}=y_r$, where r is drawn randomly from all the remaining $(n-1)$ effort values (i.e. $r\in\{1,\cdots,n\}\bigwedge\{r\neq t\}$) with equal probability. The value of SA can be interpreted to be how much better method P is than random guessing P_0. A value close to zero is discouraging and a negative value would be even worse.

7.1.2 Metric for Prediction Interval. The performance of the PIs with CLs is typically measured by the following two metrics.

Hit rate is the most commonly used evaluation metric for PIs [36, 48, 50]. The underlying idea is that: if PIs with CL α are evaluated by T software projects, it is expected that around $\alpha\times T$ projects

have actual efforts falling inside the corresponding estimated PIs. Hit rate can be calculated by first counting the number of projects whose efforts are within the PIs, and then dividing that by the total number of projects. When the number of estimates is sufficiently large, the obtained hit rate should be around the chosen CL: when the hit rate is higher, the estimated PIs are too wide; otherwise, the estimated PIs are too narrow. However, we should note that due to the small SEE data sets, we usually do not have sufficient testing projects. Hence, hit rate may deviate from its corresponding CL although the two values should be very close in essence. Merely using hit rate as a single metric may be incapable of capturing enough uncertain performance.

Relative width is another useful performance metric for PIs [41]. The underlying idea is that: of two sets of PIs with similar hit rates, the set with the narrower intervals is more informative and indicative for a higher level of expertise or more efficient use of the uncertainty information than the wider intervals. For example, a person who is only guessing may end up with an adequate hit rate, but his/her 90% PIs are extremely wide and thus of little practical use. To compare PIs for tasks of different magnitudes, the relative width of an effort PI is defined as:

$$rWidth = \frac{upB - lowB}{|Est|}, \tag{10}$$

where $upB/lowB$ denotes the upper/lower bound of the PI, and Est is the most likely point estimation. The overall performance of uncertain prediction is measured by the average relative width across all testing projects.

Larger hit rates are more likely to be associated with wider PIs, whereas lower hit rates are more likely to be associated with narrower PIs. Therefore, if two methods have different hit rates, their relative widths are not comparable. Conversely, if two methods have the same hit rate, the one providing the narrowest relative width is more informative.

7.2 Validation Method

We apply 30 runs of 10 fold Cross-Validation (CV) to validate the performance of the investigated SEE methods. The procedure is to repeat 30 times 10-fold CV with different sample orders, in order to alleviate the impact of project orders and Bootstrap displacement. We use 10-fold CV because the SEE data are usually small, and there could be high bias if using small k (e.g. $k = 2$ in k-fold) due to the lack of training data; whereas large k, such as leave-one-out with k equal to the size of data, may result in high variance [29].

We report the results using the parameters that obtained the best performance based on the 30 runs of 10 fold CV, indicating the best performance that the investigated approaches can achieve. All analysis and statistical tests are based on the mean performance across 30 runs, each of which corresponds to one 10-fold CV. We use no further separate test sets because SEE data sets are too small. If we use a further separate test set, we will have an even smaller number of projects for training and validating (model selection). Moreover, a small test set may not represent the whole space very well, so that the evaluation of the learning model would be potentially invalid.

7.3 Point Estimation Benchmark Methods

We will evaluate the performance of SynB-RVM as a point estimator against the state-of-the-art point SEE predictors including RVM, Automatically Transformed Linear Model (ATLM) [87], k-Nearest Neighbour (k-NN), Multilayer Perceptrons (MLP), Regression Tree (RT), Support Vector Regression (SVR), Bagging with RVM (*Bag-RVM*), Bagging with ATLM (*Bag-ATLM*), Bagging with RT (*Bag-RT*), and Bagging with SVR (*Bag-SVR*). As long as the proposed method performs no worse than (hopefully better than) them in terms of point estimates, its superiority to the state-of-the-art point estimators can be justified considering the additional uncertain prediction provided.

RVM is chosen for being the baseline of the proposed method. ATLM is chosen for being a newly proposed benchmark SEE model and having been shown to performance well [87]. *K*NN is chosen for being among the most popular prediction models, and due to its simplicity and intuitive interpretation that mimics the human instinctive decision-making [51, 53, 77, 78]. Several empirical studies showed its comparable and sometimes superior performance to other SEE models [38, 51, 53, 77]. ANN has been widely used in SEE, and MLP is the most common form of ANN [30]. RT is chosen for being among the most frequently used SEE models and having potential advantage for SEE [60, 86]. SVR is designed for small data problems [23], and existing work implies that it is suitable to SEE [17, 66, 66, 72]. There are several choices for SVR kernel, and linear kernel is adopted for being a better choice for SEE [66]. Bag-RVM plays as an ensemble baseline of the proposed SynB-RVM, and Bag-ATLM is expected to perform well due to the good performance of its base model. Bag-RT has shown to be frequently among the best approaches across different data sets and rarely perform considerably worse than the best approach for any data set [60]. Bag-SVR has shown to be more accurate than those based on other base learners such as MLP [30].

One potential issue of ATLM is that it may suffer certain numerical problems while giving effort predictions for some testing projects. For instance, it may produce an extremely large or even infinite effort prediction for a testing project, which is obviously impractical. This erratic effort prediction may arise from outer-interpolating training points to predict a testing project that is isolated and very distant to any of the training projects, thus causing a very erratic prediction (e.g. very large estimated effort) and large error. The situation could be worse if (e.g.) the erratically large prediction takes place in the logarithmic effort space, which would be inverse-transformed back to the original effort space causing an even larger or infinite effort prediction.

To circumvent this numerical issue, we set up a threshold for predicted effort of ATLM at the value of 10^6. Those predictions that surpass this threshold will not take part in performance evaluation for ATLM. The assigned threshold is reasonable because the actual effort values of the investigated data sets are much smaller than it. This treatment is actually giving advantage to ATLM-related methods in the performance comparison. People can take other performance metrics such as mean logarithm absolute error to alleviate or circumvent this numerical problem as long as there is no infinite prediction.

7.4 Prediction Interval Benchmark Methods

We select three categories of PI methods in Sec. 2.3, 3.1, and 3.2 to justify the uncertain estimation of SynB-RVM. The methods in Sec. 3.3 are not included because they do not provide intervals that are specific to the project being predicted. The methods in Sec. 3.4 (except for RVM) and 3.5 are not compared because they cannot provide PIs and are thus out of the scope of this paper. Our implementations for the three groups of uncertain effort methods are described in the following subsections. We also briefly describe the involved point estimators.

7.4.1 RVM. As our method uses RVM as its base leaner and RVMs have shown competitive performance against state-of-the-art point estimators [79], we compare our proposed methods against RVM in terms of both point and uncertain predictions.

7.4.2 RVM-/ATLM-based Bootstrap Wrapped PIs. We detail our implementation and denote RVM-based and ATLM-based Bootstrap wrapped methods by *BtstrpRVM* and *BtstrpATLM* respectively.

We choose Laqrichi et al. [52]'s method as the implementation for this category because it does not present the issues of other Bootstrap methods: it provides PIs and not CIs as [2], and it does not require expert knowledge as Klas et al. [50]'s method. Our implementation follows the same procedures as [52] except that the base learner we use is RVM or ATLM instead of MLP. RVM is

used for a fair comparison with the proposed RVM-based method. The replacement with RVM may even improve the performance because RVM has been shown to outperform MLP for point estimates [79]. ATLM is used so that BtstrpATLM plays as a baseline for uncertain prediction.

Our implementation for Laqrichi et al. [52]'s method is as follows: (1) Generate a number of M training sets via Bootstrap resampling, where M is a parameter of this baseline method. (2) For each resampled training set, build a RVM/ATLM model and only consider their point predictions. Hereafter, we have M trained RVM/ATLM models. (3) For estimating a testing project, we can get M point estimates from the RVM/ATLM models. (4) The point estimates produced by previous step compose a distribution that can be used to compute the PI for the testing project corresponding to the CL α. The PI is given by $[e_{(1-\alpha)/2}, e_{(1+\alpha)/2}]$, where $e_{(1-\alpha)/2}$ and $e_{(1+\alpha)/2}$ are the effort values corresponding to the $\frac{1-\alpha}{2}$-th and the $\frac{1+\alpha}{2}$-th percentiles of the distribution respectively.

7.4.3 RVM-/ATLM-based Empirical Error PIs. We detail our implementation and denote RVM-based and ATLM-based empirical error-based methods by *EmpRVM* and *EmpATLM* respectively.

We follow the procedures for the empirical version of methods in [40] except that: (a) the base learner we use is RVM/ATLM instead of a simple multivariate linear regression (MLR) model, and (b) we use all training data for a project prediction as the training set is too small after further selection. EmpRVM is built up for a fair comparison with our proposed RVM-based method, and EmpATLM plays as a baseline for uncertain prediction. The replacement with ATLM over MLR can even improve the point performance because ATLM has been shown to outperform MLR [87].

Our implementations for uncertain prediction work as follows: (1) Build the RVM/ATLM model based on all training data. (2) Calculate the empirical training error distribution measured by Balanced Relative Error (BRE) that is defined for each training sample as:

$$BRE = \begin{cases} (Act - Est)/Act, & Act \leq Est, \\ (Act - Est)/Est, & Act > Est, \end{cases} \tag{11}$$

where Act/Est is the actual/estimated effort of the training project. Here, we have a number of BREs, each of which corresponds to one training project. (3) Calculate the empirical distribution of the training errors based on α-percentiles of those BREs, where α is the corresponding CL. Specifically, we use the percentile $(1-\alpha)/2$ as the minimum BRE value and the percentile $(1+\alpha)/2$ as the maximum BRE value. (4) The lower bound (lowB) and upper bound (upB) of effort PI with CL α for a testing project are calculated using the decided minimum and maximum BRE values as:

$$lowB/upB = \begin{cases} Est' \div (1 - BRE), & BRE \leq 0 \\ Est' \times (1 + BRE), & BRE > 0 \end{cases} \tag{12}$$

respectively. Here, Est' is the point prediction of the testing project, which is determined by the trained RVM/ATLM model.

It is noteworthy that as a point estimator, the predictions from EmpRVM/EmpATLM equal to the ones from RVM/ATLM precisely. They differ only in the ways/capabilities of constructing PIs.

7.5 Parameter Settings

The parameter values of the methods investigated in this paper are shown in table 7. In particular, there are four tuning parameters for the proposed SynB-RVM: (1) the basis width c in RVM, (2) the number of Bootstrap bags M, (3) the degree of displacement in synthetic project generation ρ in Sec. 4.2, and (4) the pruning rate τ in Sec. 5.2. For RVM, its parameter c has been chosen from the values counting from 0.1 to 15 with step 0.2 (i.e. $\{0.1 : 0.2 : 15\}$). After deciding the parameter c for RVM, other investigated RVM-related methods have their specific parameters tuned based on grid search while keeping c fixed, since they can be considered as possible ways of improving their baseline performance of RVM. ATLM and EmpATLM do not have tuning parameters. For RT, the

Table 7. Parameter values of the investigated methods. The integers in parentheses are the number of investigating parameter values. The number of parameter tunings is designed to be similar among base learners, and to have three values for each tuning parameter of the ensemble methods.

Approach	Parameters
RVM	c (width) = $\{0.1 : 0.2 : 15\}$ (#75)
BtstrpRVM	c: Use the optimal c from RVM M (Bootstrap bags) = $\{30, 50, 80\}$ (#3)
BtstrpATLM EmpRVM	M (Bootstrap bags) = $\{30, 50, 80\}$ (#3) c: Use the optimal c from RVM
SynB-RVM	c: Use the optimal c from RVM M (Bootstrap bags) = $\{30, 50, 80\}$ (#3) ρ (synthetic displace) = $\{0.01, 0.1, 0.3\}$ (#3) τ (prune rate) = $\{0, 0.1, 0.2\}$ (#3)
k-NN	k (#neighbour) = $\{1 : 1 : 75\}$ (#75)
RT	L (max tree depth) = $\{-1, 2, 6, 10\}$ (#4) M (min #node per leaf) = $\{1, 2, 4, 6\}$ (#4) E (stopping error) = $\{0.0001, 0.001, 0.01, 0.1, 0.5\}$ (#5)
SVR	*kernel* = 'linear' C (regularization) = $\{0.01, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}$ (#9) ϵ (slack variables) = $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 1\}$ (#8)
MLP	L (learning rate) = $\{0.1, 0.3, 0.5\}$ (#3) M (Momentum) = $\{0.1, 0.3, 0.5\}$ (#3) N (#epochs) = $\{100, 500, 1000\}$ (#3) H (#hidden nodes) = $\{1, 5, 9\}$ (#3)
Bagging with RVM, ATLM, RT and SVR	M (Bootstrap bags) = $\{30, 50, 80\}$ (#3) Use the best parameter setting of the base learners

maximum tree depth -1 means the unlimited tree depth. For SVR, the conventional settings for regularization parameter C and slack variable ϵ are used [17, 62].

For a fair comparison, the number of parameter settings of the base learners is similar to that of RVM. Similar to those RVM-based methods, Bagging with RVM, ATLM, RT and SVR have their specific parameters tuned based on grid search while keeping the optimal parameter settings of the based learners fixed.

8 EVALUATION OF THE PROPOSED SYNB-RVM

This section aims to evaluate our method in comparison with SEE methods that have shown to perform well as described in Sec. 7.3 and 7.4. Hereafter, the three versions of our method discussed in Sec. 5.3 are denoted as *SynB-RVM_SpMn*, *SynB-RVM_1Dhist* and *SynB-RVM_2Dhist* respectively.

8.1 Evaluation of Point Estimates

This subsection aims to answer RQ1: How well can the proposed method perform in terms of point estimation? It is noteworthy that we actually have four RVM-related comparisons including RVM, EmpRVM, BtstrpRVM, and Bag-RVM, and that similarly we have four ATLM-related comparisons including ATLM, EmpATLM, BtstrpATLM, and Bag-ATLM. However, the point estimates of EmpRVM/EmpATLM/BtstrpRVM/BtstrpATLM equal to the ones of RVM/ATLM/Bag-RVM/Bag-ATLM.

Tables 8(a)~8(d) list the performance measured in MAE, MdAE, LSD and SA. We perform Friedman tests [19] for statistical comparisons of all methods across all data sets. The null hypothesis (H0) states that all methods are equivalent in terms of point prediction performance. The alternative

Table 8. Point prediction performance of the investigated methods in terms of MAE, MdAE, LSD and SA. The reported values are the mean of 30 runs of 10-fold CV. The first three columns correspond the three versions of our method. The ranks of predictors at each data set are in parentheses, and the last row lists their average rank ± std, where significant difference of Friedman tests across all data sets is highlighted in yellow (light grey). Effect size across 30 runs of each data set against the control method is also computed. SynB-RVM_2Dhist is chosen as the control method as often having the best average rank among the three versions. Cells in green (light grey)/orange (dark grey) indicate better or worse performance in the control method with medium/large effect size.

(a) Point performance of all investigated methods in terms of MAE.

Data Set	SynB-RVM _SpMn	SynB-RVM _1Dhist	SynB-RVM _2Dhist	RVM (EmpRVM)	BtstrpRVM (Bag-RVM)	ATLM (EmpATLM)	BtstrpATLM (Bag-ATLM)	kNN	SVR	MLP	RT	Bag-RT	Bag-SVR
Maxwell	3964.0(2)	3965.3(3)	3949.2(1)	4193.4(6)	4067.3(4)	4085.9(5)	6644.5(13)	4325.0(8)	5360.0(11)	6436.5(12)	4196.5(7)	4404.7(9)	5315.9(10)
Kitchenham	1682.6(5)	1674.6(4)	1668.6(3)	1725.9(6)	1779.1(7)	1603.6(2)	1389.7(1)	1784.2(8)	2230.2(11)	2399.7(13)	1989.8(10)	1879.3(9)	2231.5(12)
Cocomo81	553.4(3)	553.9(4)	554.1(5)	569.7(10)	566.9(7)	254.7(1)	277.2(2)	596.5(12)	568.6(8)	744.9(13)	589.9(11)	564.3(6)	568.8(9)
Nasa93	339.4(3)	340.7(5)	340.4(4)	355.5(6)	332.8(2)	271.4(1)	447.7(12)	438.6(11)	369.3(7)	602.8(13)	414.6(10)	376.1(9)	375.8(8)
Org1	3087.7(7)	3059.1(6)	2941.4(4)	2868.7(3)	3043.6(5)	2633.6(1)	2675.2(2)	3104.1(8)	3323.9(11)	3561.2(13)	3370.9(12)	3215.7(9)	3321.3(10)
Org2	1687.1(1)	1689.2(2)	1696.7(3)	1728.2(6)	1726.6(5)	1718.3(4)	1746.5(7)	1773.3(8)	1833.3(9)	2134.5(13)	2004.6(12)	1998.5(11)	1856.4(10)
Org3	1023.7(7)	1051.9(8)	1067.0(9)	1005.7(5)	1084.0(10)	952.4(1)	954.9(2)	993.4(3)	1208.2(11)	1543.6(13)	999.3(4)	1008.1(6)	1211.7(12)
Org4	3685.7(3)	3662.9(2)	3654.6(1)	3689.0(4)	3769.3(5)	3808.7(7)	3910.5(8)	3788.8(6)	4109.0(10)	4657.9(13)	4394.7(12)	4293.6(11)	4094.9(9)
Org5	5488.6(8)	5252.2(6)	5183.3(5)	5862.7(9)	5876.9(10)	3734.7(2)	3057.4(1)	5390.8(7)	6411.6(11)	7533.4(13)	4725.4(4)	4681.8(3)	6705.2(12)
Org6	2772.2(8)	2697.1(4)	2701.5(5)	2741.1(7)	2958.3(9)	4731.1(13)	2718.2(6)	2541.4(3)	3505.1(10)	3527.2(11)	2508.2(1)	2525.4(2)	3656.3(12)
Org7	4835.9(7)	4829.5(6)	4846.8(8)	4897.4(10)	4951.6(11)	4378.8(2)	4583.9(4)	4007.3(1)	4745.7(5)	5192.9(13)	4467.9(3)	4877.5(9)	5140.1(12)
aveRank	4.91±2.59	4.55±1.86	4.36±2.50	6.55±2.30	6.82±2.89	3.55±3.70	5.27±4.31	6.82±3.37	9.45±2.02	12.73±0.65	7.82±4.14	7.64±3.01	10.55±1.51

(b) Point performance of all investigated methods in terms of MdAE.

Data Set	SynB-RVM _SpMn	SynB-RVM _1Dhist	SynB-RVM _2Dhist	RVM (EmpRVM)	BtstrpRVM (Bag-RVM)	ATLM (EmpATLM)	BtstrpATLM (Bag-ATLM)	kNN	SVR	MLP	RT	Bag-RT	Bag-SVR
Maxwell	2104.4(1)	2105.3(2)	2109.6(3)	2176.6(6)	2220.0(8)	2116.1(4)	2259.2(9)	2146.0(5)	3022.6(12)	3499.2(13)	2180.3(7)	2602.4(10)	3005.9(11)
Kitchenham	527.2(2)	529.0(4)	528.5(3)	643.3(8)	717.9(10)	586.7(6)	575.4(5)	473.1(1)	807.2(11)	856.1(13)	623.0(7)	656.2(9)	837.7(12)
Cocomo81	77.1(8)	76.2(6)	76.1(5)	91.9(10)	76.5(7)	33.3(1)	37.1(2)	74.6(4)	78.8(9)	172.7(13)	67.2(3)	109.8(11)	121.0(12)
Nasa93	96.3(6)	97.4(8)	96.9(7)	100.1(9)	86.5(4)	66.8(1)	68.3(2)	147.3(12)	111.1(10)	283.0(13)	84.7(3)	95.6(5)	115.9(11)
Org1	603.8(11)	602.3(9)	603.8(10)	574.5(6)	532.3(4)	488.5(2)	492.0(3)	448.7(1)	546.3(5)	712.4(13)	600.2(8)	645.4(12)	595.2(7)
Org2	674.0(2)	679.4(4)	687.2(5)	624.8(1)	675.4(3)	797.3(6)	831.9(9)	813.2(7)	828.4(8)	1015.4(11)	1052.6(12)	1219.5(13)	908.9(10)
Org3	414.7(5)	414.9(6)	413.9(4)	460.6(8)	559.0(12)	405.7(1)	407.6(2)	410.7(3)	517.6(10)	645.7(13)	458.2(7)	489.7(9)	522.9(11)
Org4	1968.0(5)	1968.6(7)	1968.6(6)	2080.8(9)	2031.3(8)	1573.7(2)	1591.7(3)	1441.5(1)	2118.1(11)	1938.2(4)	2145.9(12)	2310.1(13)	2081.3(10)
Org5	3189.2(9)	3120.8(7)	3173.3(8)	2616.5(6)	3319.3(10)	2078.0(4)	1702.7(1)	2460.9(5)	3955.1(13)	3667.9(11)	1790.5(3)	1751.6(2)	3708.1(12)
Org6	1629.3(7)	1642.3(9)	1641.4(8)	1478.4(5)	1683.7(10)	1121.7(1)	1229.2(2)	1600.2(6)	2032.2(12)	1728.7(11)	1330.1(3)	1464.0(4)	2358.8(13)
Org7	4454.8(11)	4436.6(9)	4452.0(10)	4612.1(13)	4560.7(12)	2861.1(3)	2931.3(4)	2371.7(1)	3807.7(7)	3393.8(5)	2545.6(2)	3617.0(6)	4028.6(8)
aveRank	6.09±3.51	6.45±2.34	6.27±2.53	7.36±3.11	8.00±3.19	2.82±1.94	3.82±2.79	4.18±3.40	9.82±2.40	10.91±3.30	6.09±3.62	8.55±3.78	10.64±1.80

(c) Point performance of all investigated methods in terms of LSD.

Data Set	SynB-RVM _SpMn	SynB-RVM _1Dhist	SynB-RVM _2Dhist	RVM (EmpRVM)	BtstrpRVM (Bag-RVM)	ATLM (EmpATLM)	BtstrpATLM (Bag-ATLM)	kNN	SVR	MLP	RT	Bag-RT	Bag-SVR
Maxwell	0.6846(4)	0.6820(3)	0.6814(2)	0.8125(7)	0.8416(8)	0.7711(6)	0.9291(10)	0.8861(9)	1.0263(12)	1.8773(13)	0.6525(1)	0.6910(5)	1.0178(11)
Kitchenham	0.6462(4)	0.6439(2)	0.6434(1)	0.6460(3)	1.3559(13)	0.6602(6)	0.6500(5)	0.7368(8)	1.0394(12)	1.0211(10)	0.7580(9)	0.7081(7)	1.0351(11)
Cocomo81	1.6187(7)	1.6206(8)	1.6207(9)	1.9450(11)	1.8043(10)	0.5473(2)	0.5456(1)	2.0354(12)	1.5720(6)	2.1404(13)	1.2296(4)	1.0069(3)	1.4570(5)
Nasa93	0.8633(4)	0.8674(6)	0.8666(5)	1.1090(11)	0.9291(7)	0.7153(2)	0.6928(1)	1.4506(12)	0.9624(9)	2.1780(13)	0.9313(8)	0.8337(3)	0.9779(10)
Org1	0.9128(6)	0.8956(4)	0.8889(3)	1.0030(9)	0.9523(8)	0.8748(2)	0.8678(1)	1.0312(10)	1.2370(12)	1.5076(13)	0.9116(5)	0.9452(7)	1.2264(11)
Org2	0.7650(6)	0.7658(7)	0.7685(8)	0.7776(9)	0.8106(10)	0.6783(3)	0.6704(2)	0.6949(5)	0.8765(12)	0.8781(13)	0.6842(4)	0.6612(1)	0.8633(11)
Org3	0.7399(2)	0.7435(3)	0.7455(4)	0.7390(1)	2.1272(13)	0.7625(6)	0.7586(5)	0.8237(9)	1.0330(10)	1.2311(12)	0.7773(7)	0.7781(8)	1.0358(11)
Org4	0.8858(7)	0.8836(5)	0.8842(6)	0.9203(10)	0.8936(8)	0.8422(1)	0.8468(2)	0.9159(9)	1.0779(13)	1.0610(11)	0.8712(4)	0.8517(3)	1.0768(12)
Org5	0.9804(8)	0.9574(7)	0.9430(5)	1.0931(10)	1.0772(9)	0.8194(3)	0.8565(4)	0.9544(6)	1.1288(11)	1.7135(13)	0.7924(1)	0.8173(2)	1.2279(12)
Org6	0.9510(8)	0.9230(7)	0.9058(6)	0.9886(9)	1.0039(10)	0.8178(2)	0.7994(1)	0.8397(3)	1.2471(12)	1.2020(11)	0.8464(4)	0.8491(5)	1.2967(13)
Org7	0.9539(4)	0.9513(3)	0.9512(2)	0.9487(1)	1.1148(11)	1.0220(9)	0.9841(6)	0.9699(5)	1.0637(10)	1.1437(13)	0.9857(7)	0.9972(8)	1.1280(12)
aveRank	5.45±1.97	5.00±2.10	4.64±2.54	7.36±3.85	9.73±2.00	3.82±2.52	3.45±2.88	8.00±2.93	10.82±1.99	12.27±1.10	4.91±2.63	4.73±2.49	10.82±2.09

(d) Point performance of all investigated methods in terms of SA.

Data Set	SynB-RVM _SpMn	SynB-RVM _1Dhist	SynB-RVM _2Dhist	RVM (EmpRVM)	BtstrpRVM (Bag-RVM)	ATLM (EmpATLM)	BtstrpATLM (Bag-ATLM)	kNN	SVR	MLP	RT	Bag-RT	Bag-SVR
Maxwell	0.5458(2)	0.5457(3)	0.5475(1)	0.5164(6)	0.5340(4)	0.5289(5)	0.0382(13)	0.5002(8)	0.3816(11)	0.2626(12)	0.5153(7)	0.4953(9)	0.3909(10)
Kitchenham	0.5500(4)	0.5526(3)	0.5540(2)	0.5435(5)	0.3834(12)	0.5750(1)	0.5327(6)	0.5259(7)	0.4090(10)	0.3572(13)	0.4727(9)	0.4966(8)	0.4022(11)
Cocomo81	0.4954(3)	0.4949(4)	0.4948(5)	0.4833(8)	0.4831(9)	0.7692(1)	0.7473(2)	0.4590(12)	0.4841(7)	0.3208(13)	0.4650(11)	0.4854(6)	0.4814(10)
Nasa93	0.5940(3)	0.5924(5)	0.5929(4)	0.5778(6)	0.6019(2)	0.6768(1)	0.4644(12)	0.4784(11)	0.5610(7)	0.2789(13)	0.5068(10)	0.5502(9)	0.5504(8)
Org1	0.5004(7)	0.5051(6)	0.5241(4)	0.5398(3)	0.5076(5)	0.5751(1)	0.5672(2)	0.4999(8)	0.4653(10)	0.4238(13)	0.4579(12)	0.4797(9)	0.4626(11)
Org2	0.4156(1)	0.4149(2)	0.4123(3)	0.3993(6)	0.4020(5)	0.4031(4)	0.3951(7)	0.3820(8)	0.3625(9)	0.2607(13)	0.3024(12)	0.3078(11)	0.3570(10)
Org3	0.5413(7)	0.5287(8)	0.5219(9)	0.5514(5)	-1.2072(13)	0.5752(1)	0.5722(2)	0.5563(3)	0.4608(10)	0.3084(12)	0.5538(4)	0.5483(6)	0.4571(11)
Org4	0.4295(4)	0.4330(2)	0.4343(1)	0.4301(3)	0.4165(5)	0.4103(7)	0.3947(8)	0.4132(6)	0.3640(10)	0.2790(13)	0.3200(12)	0.3354(11)	0.3661(9)
Org5	0.4207(7)	0.4456(5)	0.4529(4)	0.3562(8)	0.3264(9)	0.5021(3)	-0.1153(13)	0.4303(6)	0.3245(10)	0.2049(12)	0.5027(2)	0.5058(1)	0.2923(11)
Org6	0.4662(7)	0.4806(4)	0.4798(5)	0.4736(6)	0.4279(8)	0.0890(12)	-0.1062(13)	0.5114(3)	0.3270(9)	0.3207(10)	0.5181(1)	0.5137(2)	0.2959(11)
Org7	0.2370(7)	0.2380(6)	0.2353(8)	0.2233(10)	0.1030(13)	0.3043(2)	0.2582(4)	0.3636(1)	0.2471(5)	0.1807(12)	0.2898(3)	0.2305(9)	0.1890(11)
aveRank	4.73±2.33	4.36±1.86	4.18±2.56	6.00±2.10	7.73±3.82	3.45±3.47	7.45±4.66	6.64±3.35	8.91±1.81	12.36±0.92	7.55±4.32	7.36±3.32	10.27±1.01

hypothesis (H1) states that at least one pair of methods differs. The Friedman test with the significance level 0.05 rejects H0 with the p-value of 3.11×10^{-13}, 4.11×10^{-10}, 0 and 3.38×10^{-11} for MAE, MdAE, LSD and SA respectively.

Friedman tests also provide rankings of the methods. Let $r_j^{(i)}$ be the rank of the j-th method on the i-th data set, and N be the number of data sets. The average rank of method j is calculated as:

$$R_j = \frac{1}{N} \sum_i r_j^{(i)}. \tag{13}$$

The average ranks can provide a reasonable idea of how the methods compare to each other given rejection of the null hypothesis [19]. In table 8, the integer in parentheses along with a method is its rank over all methods on each data set ($r_j^{(i)}$), and the last row lists the average rank of each method across all data sets (R_j). We can see that the three versions of SynB-RVM can usually outperform the other RVM-related methods. Specifically, when measured in MAE, SynB-RVM_2Dhist achieves the best average rank, SynB-RVM_1Dhist performs the second followed by SynB-RVM_SpMn with slightly worse average rank, and BtstrpRVM (and Bag-RVM) performs the worst among all RVM-related methods. ATLM (and EmpATLM) can always have slightly better average rank than SynB-RVM, and the performance of BtstrpATLM (and Bag-ATLM) shifts according to the performance metrics. Nevertheless, SynB-RVM usually significantly outperforms other methods in at least one metric.

Next, we conduct post-hoc tests for a more formal comparison. SynB-RVM_2Dhist is chosen as the control method for often performing the best among the three versions of the proposed method. For each data set, we also compute the effect size across the 30 runs against SynB-RVM_2Dhist and highlight the difference with medium/large magnitude. Effect size is a simple way of quantifying the differences between two methods with multiple runs [85]. The Vargha and Delaney's A_{12} is adopted for being a non-parametric effect size and making no assumptions on the underlying distribution [4, 85]. It is interpreted, according to Vargha and Delaney's categories [85], as: small ($\geq$0.56), medium ($\geq$0.64) and large ($\geq$ 0.71). The results are as follows:

• In terms of MAE, post-hoc tests with Holm-Bonferroni corrections for comparing each method against SynB-RVM_2Dhist detect significant superiority to SVR, MLP, RT and Bag-SVR. No significant difference can be found with respect to the three versions of SynB-RVM or to the RVM/ATLM-related methods. SynB-RVM_2Dhist has superiority to RVM (EmpRVM), BtstrpRVM (Bag-RVM) and BtstrpATLM (Bag-ATLM) in most data sets with medium/large effect size, but performs worse than ATLM (EmpATLM) in many data sets with medium/large effect size.

• In terms of MdAE, post-hoc tests detect that SynB-RVM_2Dhist performs significantly better than RVM (EmpRVM), BstrpRVM (Bag-RVM), SVR, MLP, Bag-RT and Bag-SVR. No significant difference can be found among the three versions of SynB-RVM. RT and kNN have better average ranks in terms of MdAE than in terms of MAE, and perform similarly to SynB-RVM_2Dhist. ATLM (EmpATLM) and BtstrpATLM (Bag-ATLM) have similar overall performance, but outperform our method in many data sets with medium/large effect size.

• In terms of LSD, post-hoc tests detect that SynB-RVM_2Dhist performs significantly better than BtstrpRVM (Bag-RVM), kNN, SVR, MLP and Bag-SVR. No significant difference can be found among the three versions of SynB-RVM. ATLM (EmpATLM) and BtstrpATLM (Bag-ATLM) have similar overall performance, and are superior or inferior to SynB-RVM_2Dhist with medium/large effect size depending on the data sets.

• In terms of SA, post-hoc tests detect detect that SynB-RVM_2Dhist performs significantly better than BtstrpRVM (Bag-RVM), SVR, MLP and Bag-SVR. No significant difference can be found among the three versions of SynB-RVM. Similar to MAE, SynB-RVM_2Dhist is superior to RVM (EmpRVM),

BtstrpRVM (Bag-RVM) and BtstrpATLM (Bag-ATLM) in most data sets with medium/large effect size, but performs worse than ATLM (EmpATLM) in many data sets with medium/large effect size.

Overall, the performance comparisons slightly vary in terms of different metrics, consistent with the observations in [61]. ATLM (EmpATLM) can outperform SynB-RVM with medium/large effect size in many data sets, but their performance is statistically similar across data sets. Nevertheless, our method is more consistently among the best methods regardless of the metrics.

Moreover, we discuss the magnitude of performance difference in terms of SA for its interpretability. We can see from table 8(d) that the magnitude of performance difference varies depending on the data sets and the competing methods. In the data sets where SynB-RVM performs worse with medium/large effect size, the magnitude is usually of little practical significance except for EmpATLM and BtstrpATLM on Cocomo81, where SynB-RVM performs worse than EmpATLM/BtstrpATLM with 0.49 vs 0.76/0.74. In the data sets where SynB-RVM performs better with medium/large effect size, the magnitude can be very large in Maxwell against BtstrpATLM with 0.54 vs 0.038, in Org3 against BtstrpRVM with 0.52 vs -1.207, in Org5 against BtstrpATLM with 0.45 vs -0.113, and in Org6 against EmpATLM with 0.47 vs 0.089 and against BtstrpATLM with 0.47 vs -0.106. The performance improvement of SynB-RVM over RVM is usually small, but it can achieve better relative width as will be discussed in section 8.2.2. Overall, these results suggest that SynB-RVM is more likely to perform better and sometimes much better in practice.

We should note one limitation of ATLM, as discussed in Sec. 7.3, ATLM-related methods may suffer certain numerical problems when giving effort prediction for some testing project that is very distant form any of the training points. We circumvent it by setting up a reasonable threshold, surpassing which the predicted effort will not take part in the final prediction calculation (for BtstrpATML and Bag-ATLM) and performance evaluation (for ATLM and EmpATLM).

An interesting observation is that ATLM-related methods can largely outperform all the others in some data sets such as Cocomo81, Nasa93 and Org5. It suggests fairly good multiple linear fittings between the transformed input features and the transformed efforts. The variable transformation includes *logarithm*, *square-root* and *none* [87]. Pearson correlations between the logarithm of size-related feature (e.g. line of codes) and logarithm of effort attain fairly large values at 0.8466, 0.8435, and 0.8236 for the listed data sets respectively, suggesting good linear fittings between inputs and outputs after proper data transformation. This results also confirm the arguments from the paper [44] saying that "with appropriate transformations, multiple linear regression can produce suitable and accurate predictive models".

In summary, the experimental results suggest that our method significantly improves over RVM (EmpRVM) and BtstrpRVM (Bag-RVM), but performs similar to ATLM (EmpATLM) and BtstrpATLM (Bag-ATLM). Nevertheless, SynB-RVM still holds its merits over ATLM on its capability of making uncertain prediction and absence of numerical problems. Our results confirm that our method is able to improve its base learner for better point estimates, and can outperform the state-of-the-art point estimators significantly. It also confirms the results from the paper proposing ATLM, which shows this baseline SEE model to be competitive against state-of-the-art effort estimators [87].

8.2 Evaluation on Prediction Intervals with Confidence Levels

This subsection aims to answer RQ2: Can SynB-RVM derive the PIs with good hit rate and narrower PIs compared with other PI methods? We will answer RQ2 in two parts: (1) Can the proposed PIs achieve good hit rates? (2) Can the proposed PIs obtain small relative width with similar hit rate? Evaluation of uncertain prediction should be based on the best parameter settings with respective to some metric. In this sense, we can choose the best parameter settings according to hit rates,

relative widths, MAE, MdAE, LSD or SA. In this paper, the best parameter settings are decided in accordance with the best MAE for reflecting the actual deviation of the estimated efforts.

8.2.1 Evaluation on Hit Rate. To evaluate the PIs in terms of hit rate, we produce PIs with 12 CLs at $\{10, 20, \cdots, 90, 68.27, 95.45, 99.73\}$% according to Eq. (7) and (8). CLs at $\{68.27, 95.45, 99.73\}$% are included due to their easy computation by setting $j = \{1, 2, 3\}$ in Eq. (8).

In practice, hit rate that is either equal to or greater than its CL is considered to be satisfactory. When the hit rates are smaller than their CLs, the method fails in terms of achieving the required hit rate. In this case, the smaller the hit rate the worse the performance of the method. When the hit rates are equal or greater than their CLs, this means that the method succeed in reaching the required hit rate. If the hit rates surpass their CLs, this does not mean that the method is unsuccessful in achieving the required hit rate. In formula, the loss function of hit rate is defined as

$$\mathscr{L}(h) = \begin{cases} cl - h, & cl > h \\ 0, & cl \leq h \end{cases} \tag{14}$$

where h is the actual hit rate and cl is the corresponding CL. When the hit rate equals to or surpasses its CL, the loss is zero; when the hit rate is lower than its CL, the loss equals to their distance.

For an idea of the achievable hit rates of the uncertain methods, table 9(a) lists the median hit rates across 11 data sets for each method at each CL. The values in parentheses are the percentages (in 100%) of data sets that succeed in reaching the desired hit rates. We can see that our method can usually succeed in reaching the required hit rates, while most others fail in reaching them. In particular, BtstrpRVM/BtstrpATLM always has much lower hit rates than required. Though having better hit rates than BtstrpRVM/BtstrpATLM, EmpRVM/EmpATLM still rarely succeed in reaching the CLs, i.e., the percentage of success is almost always zero. The magnitude of superiority of SynB-RVM in terms of hit rate is usually large compared with all the other methods except for RVM. Taking CL 80% as an example, SynB-RVM_1Dhist can achieve 81.6% hit rates that is superior to the best 70.3% hit rate achieved by EmpATLM. The difference between SynB-RVM and RVM in terms of hit rate is usually small. SynB-RVM usually surpasses the CLs, meaning that adjusting the method to reduce its hit rate could potentially help improving the width of the PIs produced by this method. A possible future enhancement could be to provide a non-symmetric interval prediction as discussed in Sec. 11 ("*Non-symmetric effort prediction uncertainty*").

Table 9(b) lists the average rank of each method across all data sets in terms of hit rate. To perform thorough comparisons, for each of the 12 CLs, we conduct one Friedman test with the significance level 0.05 on the hit rates. The null hypothesis (H0) states that the hit rates of the investigated methods are equivalent across data sets. The alternative hypothesis (H1) states that at least one pair of methods differs in terms of hit rate. All the 12 Friedman tests reject H0 with very small p-values ranging from 4.0301×10^{-14} to 1.1826×10^{-4}. After that, we conduct post-hoc tests with Holm-Bonferroni corrections for each CL. Positive/negative signs in the parentheses denote significant difference/none-difference against the control methods that have $\star$ in their parentheses. The control methods may vary for different CLs and are chosen for having the best average ranks. For instance, SynB-RVM_1Dhist has the best average rank for $CL_{10\%}$ and was chosen as the control method; whereas RVM has the best average rank for $CL_{50\%}$ and was chosen as the control method.

We can see from table 9(b) that no significant difference has been found between the control methods and RVM and the three versions of SynB-RVM based on the post-hoc tests with Holm-Bonferroni corrections with the significance level 0.05 (see the negative signs associated to them). Actually, the control methods are either RVM or SynB-RVM over all CLs except for CL 0.997 (see the star signs). In contrast, all PIs derived from BtstrpRVM perform significantly worse than the ones from the control methods. One possible reason for their worse performance than RVM may be

Table 9. Performance evaluation of the uncertain methods in terms of hit rates measured in Eq. (14).

(a) Median hit rates across 11 data sets for each method at each CL. The values in the parentheses are the percentages (in 100%) of data sets that succeed in hit rates. Cells in yellow (light grey) highlight methods whose medians were successful in reaching or surpassing the corresponding hit rate.

CL%	RVM	BtstrpRVM	EmpRVM	BtstrpATLM	EmpATLM	#_SpMn	#_1Dhist	#_2Dhist
10.00	16.6(72.7)	4.4(0.0)	7.7(0.0)	2.5(0.0)	8.4(0.0)	12.5(81.8)	12.4(81.8)	12.5(81.8)
20.00	36.9(63.6)	9.1(0.0)	16.1(9.1)	5.7(0.0)	16.9(9.1)	24.8(72.7)	24.8(72.7)	24.7(81.8)
30.00	50.7(63.6)	14.7(0.0)	24.9(0.0)	8.8(0.0)	26.2(0.0)	39.5(63.6)	39.2(63.6)	39.2(63.6)
40.00	56.6(63.6)	19.8(0.0)	32.8(0.0)	14.4(0.0)	34.8(0.0)	50.5(63.6)	51.7(63.6)	51.6(63.6)
50.00	61.9(72.7)	24.6(0.0)	43.2(0.0)	18.8(0.0)	44.9(0.0)	57.0(54.5)	59.5(54.5)	62.5(54.5)
60.00	66.8(63.6)	29.8(0.0)	53.8(0.0)	23.2(0.0)	52.5(0.0)	67.0(54.5)	68.8(54.5)	74.8(54.5)
68.27	75.4(72.7)	31.9(0.0)	59.5(0.0)	25.0(0.0)	61.4(0.0)	77.1(63.6)	75.1(63.6)	78.6(63.6)
70.00	76.4(72.7)	33.8(0.0)	59.5(0.0)	26.6(0.0)	61.9(0.0)	78.3(63.6)	77.6(63.6)	79.8(63.6)
80.00	81.0(63.6)	40.5(0.0)	69.7(0.0)	31.6(0.0)	70.3(0.0)	81.3(63.6)	81.6(63.6)	82.2(63.6)
90.00	85.9(36.4)	50.9(0.0)	79.5(0.0)	41.1(0.0)	83.8(0.0)	86.3(27.3)	86.0(27.3)	85.9(27.3)
95.45	89.0(9.1)	53.0(0.0)	86.2(0.0)	45.4(0.0)	89.0(0.0)	88.7(9.1)	88.7(18.2)	88.8(18.2)
99.73	94.9(0.0)	61.7(0.0)	88.4(0.0)	49.1(0.0)	91.2(0.0)	93.5(0.0)	94.3(0.0)	93.4(0.0)

(b) Average ranks and statistical tests of uncertain methods across 11 data sets in terms of hit rate at each CL. Positive/Negative signs in the parentheses denote significant difference/none-difference against the control method by Friedman test with the significance level 0.05. The control methods have ⋆ in their parentheses. For significant difference, medium/large effect size against the control method across all data sets is highlighted in orange (dark grey)/yellow (light grey).

CL%	RVM	BtstrpRVM	EmpRVM	BtstrpATLM	EmpATLM	#_SpMn	#_1Dhist	#_2Dhist
10.0	2.73 (−)	6.91 (+)	5.95 (+)	6.91 (+)	5.32 (+)	2.73 (−)	2.45 (⋆)	3.00 (−)
20.0	2.77 (−)	6.95 (+)	5.86 (+)	7.09 (+)	5.18 (+)	2.68 (−)	2.86 (−)	2.59 (⋆)
30.0	2.77 (−)	6.91 (+)	5.82 (+)	7.00 (+)	4.64 (−)	2.68 (⋆)	3.23 (−)	2.95 (−)
40.0	2.68 (⋆)	6.82 (+)	5.82 (+)	6.91 (+)	5.18 (+)	2.82 (−)	2.95 (−)	2.82 (−)
50.0	2.55 (⋆)	6.91 (+)	6.00 (+)	6.82 (+)	5.00 (+)	2.82 (−)	3.09 (−)	2.82 (−)
60.0	2.45 (⋆)	7.00 (+)	5.91 (+)	6.82 (+)	4.82 (+)	3.23 (−)	2.91 (−)	2.86 (−)
68.3	2.59 (⋆)	6.91 (+)	6.09 (+)	6.55 (+)	4.73 (−)	2.95 (−)	3.32 (−)	2.86 (−)
70.0	2.68 (⋆)	6.91 (+)	6.09 (+)	6.45 (+)	4.73 (−)	2.95 (−)	3.14 (−)	3.05 (−)
80.0	2.59 (⋆)	6.73 (+)	5.91 (+)	6.64 (+)	4.45 (−)	3.05 (−)	3.50 (−)	3.14 (−)
90.0	2.77 (⋆)	6.36 (+)	5.45 (+)	6.73 (+)	3.73 (−)	3.82 (−)	3.59 (−)	3.55 (−)
95.5	3.05 (⋆)	6.82 (+)	5.18 (−)	6.45 (+)	3.45 (−)	3.95 (−)	3.59 (−)	3.50 (−)
99.7	3.64 (−)	6.91 (+)	4.45 (−)	6.36 (+)	2.73 (⋆)	4.09 (−)	3.82 (−)	4.00 (−)

the invertibility problem when training RVM with replicated projects. SynB-RVM overcomes this problem by replacing the replicated training projects with their synthetic counterparts as in Alg. 1. Similarly, post-hoc tests have found significantly worse performance for EmpRVM compared with the control methods in most hit rates. For ATLM-related methods, all PIs provided by BtstrpATLM are significantly worse than the ones from the control methods. For EmpATLM, the PIs of CLs that are equivalent or lower than 60% perform significantly worse than the ones from the control methods; post-hoc tests cannot detect significant difference for the CLs greater than 60%, but table 9(a) shows large superiority of SynB-RVM to EmpATLM for the CLs until 90%.

Though higher CL is more appealing to industry, Jørgensen et. al. suggested "*not to ask for high confidence (90 percent, or worse, 98 percent) effort prediction intervals*" because "*lower confidence intervals are much likely to be realistic*" [36]. This is because lower CLs are more likely to be achieved in practice and thus provide more precise and useful information to the PMs. Several studies in industry and academia had also showed strong bias towards over-confidence for the predicted effort PIs [36, 41, 50], where higher CLs are not really reachable in practice. Even when one can

reach a higher hit rate, the PI widths are usually too wide to be informative [2, 36, 79]. Therefore, we opt to use PIs with a lower CL such as 80% or even 60%, allowing only one/two projects to exceed the upper bound and only one/two to fall below the lower bound on average.

In summary, the experimental results and statistical tests show that SynB-RVM can usually achieve significantly better hit rates than other uncertain methods except for RVM, where they perform similarly. The performance superiority is always very large in practice especially for CLs below 90% that are more pragmatic.

8.2.2 Evaluation on Relative Width. We evaluate relative width based on the PIs generated in Sec. 8.2.1. Larger hit rates may be associated to wider PIs, whereas lower hit rates may be associated to narrower PIs. Therefore, if two methods have different hit rates, their relative widths are not comparable. Conversely, if two methods have the same hit rate, the one providing the narrower relative width is considered to be more informative. One question is then what hit rates should be selected for the comparison. Hit rates equal to or greater than the CLs are satisfactory. However, most methods were unable to reach their CLs. In the end, we fixed the hit rates to the values that are similar to the largest hit rate achievable by all methods.

We set up the following evaluation procedures to find the similar hit rates (table 10) and their corresponding relative widths (table 11): (1) For each data set, find the minimum of the highest hit rates across all methods and set this value as the benchmark hit rate denoted as B_HitR. The benchmark hit rate of each data set is chosen to be the highest hit rate that can be achievable by all uncertain methods. In other words, given the set composed by the highest hit rates achieved by the uncertain methods for a data set, the benchmark hit rate is the lowest hit rate in this set. The benchmark hit rates are reported in the second column of table 10, each corresponding to one data set. (2) For each method, find the closest hit rates to the benchmark values across all data sets and form the main body of table 10. Friedman test with the significance level 0.05 does not find significant difference on these hit rates with the p-value 0.8763, indicating the similarity of these values as desired for a fair comparison of their widths. (3) Find the relative widths in line with these hit rates and produce table 11. In this way, we can compare the relative widths with similar hit rates. Smaller values represent better exploitation of uncertainty and more informative PIs. Friedman test with the significance level 0.05 rejects null hypothesis (H0) with the p-value 2.28×10^{-4}, indicating that at least one pair of the methods differs.

Next, we conduct post-hoc tests for more detailed comparisons. We can see from table 11 that the three versions of our method can usually produce much narrower PIs compared with RVM and BtstrpRVM while reaching similar hit rates. SynB-RVM_1Dhist has the best average rank, and thus is chosen as the control method. Post-hoc tests with Holm-Bonferroni corrections have found significant difference over RVM and BtstrpRVM. No significant difference has been found over EmpRVM, BtstrpATLM or EmpATLM. Nevertheless, SynB-RVM_1Dhist has large magnitude of superiority to these methods with medium/large effect size in terms of relative width in most data sets. SynB-RVM_1Dhist performs similarly to SynB-RVM_2Dhist, but is superior to SynB-RVM_SpMn with medium/large effect size in many data sets.

In practice, SynB-RVM_1Dhist can outperform other RVM-based methods with large magnitude. For instance, SynB-RVM_1Dhist has much narrower PIs in Maxwell at 1.1964 against RVM at 5.3690 and against BtstrpRVM at 3.5965, in Cocomo81 at 1.7579 against RVM at 7.8990, against BtstrpRVM at 4.0331 and against EmpRVM at 5.9306, in Nasa93 at 2.4906 against RVM at 24.7529 and against BtstrpRVM at 8.3049. When it performs worse than some uncertain methods with medium/large effect size, the magnitude of performance inferiority is small. For ATLM-based uncertain methods, the magnitude of performance difference becomes smaller. Nevertheless, SynB-RVM still holds superiority for having better relative widths with medium/large effect size in more data sets.

Table 10. Similar hit rates of the uncertain methods. *B_HitR* denotes the benchmark hit rates, which are the minimum of the highest hit rates across all methods. The chosen actual hit rates may correspond to different CLs. The reported values are the mean of 30 runs of 10-fold CV.

Data Set	*B_HitR*	RVM	BtstrpRVM	EmpRVM	BtstrpATLM	EmpATLM	#_SpMn	#_1Dhist	#_2Dhist
Maxwell	0.7161	0.7538	0.7161	0.7177	0.6694	0.7099	0.7134	0.7161	0.7113
Kitchenham	0.3306	0.3687	0.2791	0.2874	0.3306	0.2917	0.2693	0.2630	0.2562
Cocomo81	0.6587	0.6587	0.6545	0.7228	0.6201	0.6201	0.6556	0.6524	0.6556
Nasa93	0.8032	0.7971	0.7921	0.8122	0.8416	0.8043	0.8036	0.8032	0.8036
Org1	0.4912	0.4890	0.4474	0.4592	0.4912	0.4759	0.5689	0.4105	0.4114
Org2	0.5010	0.5396	0.5094	0.4844	0.5010	0.5062	0.5333	0.5271	0.5292
Org3	0.3358	0.2957	0.3331	0.2963	0.3358	0.3805	0.5286	0.5185	0.5278
Org4	0.2858	0.2183	0.2945	0.2596	0.2858	0.2915	0.2470	0.2484	0.2481
Org5	0.6175	0.6175	0.6175	0.5794	0.6095	0.5730	0.6254	0.6286	0.6349
Org6	0.4758	0.3576	0.4455	0.4318	0.4758	0.4530	0.5515	0.5045	0.4530
Org7	0.3349	0.3190	0.3349	0.3651	0.3270	0.3397	0.3460	0.3508	0.3492

Table 11. Relative widths with similar hit rates of all uncertain methods. The reported values are the mean of 30 runs of 10-fold CV. The last row lists the average ranks in terms of better relative width, where significant difference of Friedman tests across all data sets is highlighted in yellow (light grey). Effect size across 30 runs of each data set against the control method is computed. SynB-RVM_1Dhist is chosen as the control method for having the best average rank among the three versions of SynB-RVM. Cells in green (light grey)/orange (dark grey) indicate significantly better/worse in the control method with medium/large effect size.

Data Set	RVM	BtstrpRVM	EmpRVM	BtstrpATLM	EmpATLM	#_SpMn	#_1Dhist	#_2Dhist
Maxwell	5.3690(8)	3.5965(7)	1.3463(5)	1.1753(1)	1.5491(6)	1.2105(4)	1.1964(3)	1.1920(2)
Kitchenham	0.5440(6)	0.6119(8)	0.3736(4)	0.5602(7)	0.4309(5)	0.2880(3)	0.2782(2)	0.2758(1)
Cocomo81	7.8990(8)	4.0331(6)	5.9306(7)	0.8547(1)	0.8838(2)	1.7850(5)	1.7579(4)	1.7434(3)
Nasa93	24.7529(8)	8.3049(7)	2.6709(6)	2.4244(2)	1.4536(1)	2.5382(5)	2.4906(4)	2.4635(3)
Org1	1.2073(8)	1.0177(7)	0.7698(4)	0.9296(6)	0.7343(3)	0.8693(5)	0.5790(1)	0.5807(2)
Org2	0.7568(5)	0.8353(6)	0.7420(4)	1.0378(8)	0.8598(7)	0.6395(1)	0.6397(2)	0.6451(3)
Org3	0.5582(2)	1.1071(8)	0.4536(1)	0.6992(3)	0.7540(4)	0.8216(5)	0.8444(6)	0.9103(7)
Org4	0.5955(5)	0.7351(8)	0.5452(4)	0.6661(7)	0.6005(6)	0.4627(1)	0.4630(2)	0.4635(3)
Org5	1.5556(7)	6.3404(8)	1.1957(5)	1.1817(4)	0.8633(1)	1.2568(6)	1.1067(2)	1.1165(3)
Org6	0.8125(1)	1.6334(7)	0.8477(2)	2.5781(8)	1.2453(5)	0.9132(3)	1.5098(6)	1.2052(4)
Org7	0.9659(4)	2.4802(8)	0.7741(1)	1.1289(7)	0.9961(5)	0.8943(2)	0.9441(3)	1.0276(6)
aveRank	5.64 ± 2.38	7.27 ± 0.75	3.91 ± 1.83	4.91 ± 2.64	4.09 ± 1.98	3.64 ± 1.67	3.18 ± 1.59	3.36 ± 1.67

Table 12. Relative widths with similar (and higher) hit rates of SynB-RVM_ht1D and EmpRVM. The reported values are the mean of 30 runs of 10-fold CV. Effect size across 30 runs of each data set against SynB-RVM_1Dhist is computed. Cells in green (light grey)/orange (dark grey) indicate better/worse in the control method with medium/large effect size.

Data Set	Hit rate			Relative width	
	B_HitR	EmpRVM	#_1Dhist	EmpRVM	#_1Dhist
Maxwell	0.7957	0.8446	0.7957	3.2417	1.6536
Kitchenham	0.9623	0.9623	0.9664	2.5938	4.1532
Cocomo81	0.7794	0.7794	0.7995	44.5693	2.7822
Nasa93	0.8032	0.8122	0.8032	2.6709	2.4906
Org1	0.9325	0.9408	0.9325	6.9109	4.2131
Org2	0.6542	0.6542	0.6333	1.0897	0.8198
Org3	0.9842	0.9842	0.9827	3.5895	8.3333
Org4	0.9347	0.9544	0.9347	5.9212	3.5339
Org5	0.8175	0.8175	0.8127	2.5726	1.8341
Org6	0.8682	0.8682	0.8576	1.7567	3.4535
Org7	0.8619	0.8619	0.8635	2.7669	2.3117

Table 13. Summary of performance comparisons of SynB-RVM against the other methods. The point prediction metrics include MAE, MdAE, LSD and SA, and the uncertain prediction metrics include hit rate and relative width. Equality/positive/negative sign denotes insignificantly different/significantly better/significantly worse performance of SynB-RVM against each method. Non-existing comparisons are denoted as N/A. Note that the summarized comparison in hit rate is an overall description across the 12 CLs.

PF Metric	RVM	Bag-RVM	BtstrpRVM	EmpRVM	ATLM	Bag-ATLM	BtstrpATLM	EmpATLM	kNN	SVR	MLP	RT	Bag-RT	Bag-SVR
MAE	=	=	=	=	=	=	=	=	=	+	+	+	=	+
MdAE	+	+	+	+	=	=	=	=	=	+	+	=	+	+
LSD	=	+	+	=	=	=	=	=	+	+	+	=	=	+
SA	=	+	+	=	=	=	=	=	=	+	+	=	=	+
hitR	=	N/A	+	+	N/A	N/A	+	+	N/A	N/A	N/A	N/A	N/A	N/A
rWidth	+	N/A	+	=	N/A	N/A	=	=	N/A	N/A	N/A	N/A	N/A	N/A

Evaluation of relative width with higher CLs. We compare SynB-RVM_1Dhist against EmpRVM following the same evaluation procedures in terms of relative widths. SynB-RVM_1Dhist (EmpRVM) is chosen for having the best average rank among the three versions of SynB-RVM (the competitors) according to table 11. In this manner, we can reach higher benchmark hit rates, and thus the relative widths in line with higher hit rates can be evaluated. The first part of table 12 lists the benchmark hit rates and the chosen hit rates. Wilcoxon sign-rank test with the significance level 0.05 does not find significant difference on the hit rates with p-value 0.8984, indicating the similarity of these values as desired for a fair comparison of their widths.

The second part of table 12 lists the relative widths in line with the chosen hit rates. Wilcoxon sign-rank test with the significance level 0.05 cannot reject the null hypothesis (H0) with p-value 0.3652. This means that in some datasets our approach is better, and in some others EmpRVM is better since it is a statistical test across data sets. Nevertheless, SynB-RVM_1Dhist can achieve better relative widths than EmpRVM with medium/large effect size in most data sets. The magnitude of performance superiority is often very large.

Overall, SynB-RVM produces significantly better relative width than that of RVM and BtstrpRVM across data sets, and is superior to EmpRVM, EmpATLM, and BtstrpATLM with medium/large effect size in most data sets. The superiority magnitude can be very large over other RVM-related uncertain methods in practice.

8.3 Brief Summary

The performance comparisons in terms of point and uncertain prediction are summarized in table 13, which demonstrates the superiority of SynB-RVM in terms of the overall performance.

In terms of point estimation, SynB-RVM significantly improves over RVM (EmpRVM), BtstrpRVM (Bag-RVM), kNN, SVR, MLP, RT, Bag-RT and Bag-SVR with respect to at least one metric, and performs similar to ATLM (EmpATLM) and BtstrpATLM (Bag-ATLM). In terms of hit rate, SynB-RVM can usually achieve significantly better performance over the other methods except for RVM. In terms of relative width, SynB-RVM can produce significantly better PIs than those from RVM and BtstrpRVM, and performs similarly to EmpRVM, BtstrATLM and EmpATLM. Nevertheless, SynB-RVM has large magnitude of performance superiority to EmpRVM, BtstrATLM and EmpATLM with medium/large effect size in most data sets. Altogether, SynB-RVM is a robust winner and never performs significantly worse than its competitors.

9 INVESTIGATION ON THE SYNB-RVM COMPONENTS

This section aims at answering RQ3: which components of SynB-RVM contribute to the prediction improvement over its base learner RVM in terms of both point and uncertain effort estimation? It will provide a more thorough understanding of SynB-RVM, contributing to its external validity. In particular, we will investigate the following questions: (a) Are the three methods for deriving

the probabilistic prediction similar in terms of final point and uncertain prediction? (b) Do the synthetic displacement and Bootstrap pruning of SynB-RVM contribute towards improving the final point and uncertain prediction?

9.1 Three Methods for Deriving the Final Probabilistic Prediction

To answer RQ3 (a), we can apply statistical tests on SynB-RVM_SpMn, SynB-RVM_1Dhist and SynB-RVM_2Dhist to investigate the significance of their difference. This is possible because the only difference between the three versions of SynB-RVM is the method for deriving the final probabilistic predictions according to Sec. 5.3.

For effort point estimation, Friedman test at the significance level of 0.05 was applied to the MAEs, MdAEs, LSDs and SAs of SynB-RVM_SpMn, SynB-RVM_1Dhist and SynB-RVM_2Dhist shown in table 8 respectively. The null hypothesis (H0) states that they are equivalent. The alternative hypothesis (H1) states that at least one pair of the three versions of the proposed SynB-RVM differs. No significant difference has been found with the p-value of 0.9204, 0.7145, 0.4617, and 0.9204 for MAE, MdAE, LSD and SA respectively.

Next, we consider the difference of the three versions of SynB-RVM for uncertain prediction. In term of hit rate, we conduct a Friedman test at the significance level of 0.05 on the hit rates of SynB-RVM_SpMn, SynB-RVM_1Dhist, and SynB-RVM_2Dhist shown in table 9(b) for each of the 12 CLs. For each CL, the null hypothesis (H0) is that the three versions of SynB-RVM are equivalent in terms of hit rates. The alternative hypothesis (H1) states that at least one pair of the three versions of SynB-RVM differs. No significant difference has been found for either of the 12 CLs with the p-values ranging from 0.4617 to 0.9966. In terms of relative width, Friedman test at the significance level of 0.05 was applied to the relative widths of SynB-RVM_SpMn, SynB-RVM_1Dhist, and SynB-RVM_2Dhist shown in table 11 that achieve similar hit rates. The null hypothesis (H0) states that they are equivalent in terms of relative widths. The alternative hypothesis (H1) states that at least one pair of the methods differ. No significant difference has been found with the p-value of 0.9204.

Therefore, the three versions of the proposed SynB-RVM perform similarly across data sets in both point and uncertain effort prediction.

9.2 Synthetic Displacement and Bootstrap Pruning

To answer RQ3 (b), SynB-RVM_1Dhist is compared with its variants where *synthetic displacement* and/or *Bootstrap pruning* are/is removed contributing to *#_rmAll, #_rmSyn and #_rmPru.* It enables us to explore the effectiveness of the two components in improving the performance of its base model RVM. SynB-RVM_1Dhist is chosen among the three versions for its best relative width and since they are shown to be statistically similar in Sec. 9.1. We follow the same validation design in Sec. 7 to compare the three variations against SynB-RVM_1Dhist and RVM.

9.2.1 Comparisons of Point Estimates. Table 14 shows the point performance of the five investigated methods across eleven data sets. We perform Friedman tests for the statistical comparisons. The null hypothesis (H0) states that all methods are equivalent in terms of point prediction performance. The alternative hypothesis (H1) states that at least one pair of methods differs.

Friedman tests with the significance level 0.05 reject H0 in terms of all metrics except for MdAE where H1 has to be taken. Post-hoc tests with Holm-Bonferroni corrections for each method against Syn-RVM_1Dhist detect significant difference with respect to RVM, #_rmAll, and #_rmSyn in term of MAE, LSD, and SA. No significant difference is detected with respect to #_rmPru.

Table 14. Point performance of RVM, SynB-RVM_1Dhist and its three variants. The reported values are the mean of 30 runs of 10-fold CV. The ranks for each data set are in parentheses. The last row lists the average ranks where significant difference of Friedman tests across all data sets is highlighted in yellow (light grey).

(a) MAE. Significant difference detected with p-value $4.10 * 10^{-6}$.

Data Set	RVM	#_1Dhist	#_rmAll	#_rmPru	#_rmSyn
Maxwell	4193.43(5)	3965.25(1)	3982.31(4)	3965.25(2)	3980.15(3)
Kitchenham	1725.89(3)	1674.57(1)	1753.69(4)	1674.83(2)	1753.88(5)
Cocomo81	569.72(5)	553.92(1)	564.99(3)	553.98(2)	565.02(4)
Nasa93	355.52(5)	340.72(1)	341.83(3)	340.72(2)	342.91(4)
Org1	2868.65(1)	3059.13(2)	3069.56(5)	3059.86(3)	3061.54(4)
Org2	1728.16(5)	1689.17(2)	1727.95(4)	1689.17(1)	1724.15(3)
Org3	1005.73(1)	1051.88(3)	1143.41(4)	1051.88(2)	1147.44(5)
Org4	3689.02(3)	3662.91(2)	3742.61(4)	3662.91(1)	3747.12(5)
Org5	5862.69(5)	5252.19(1)	5743.85(4)	5315.49(2)	5721.11(3)
Org6	2741.10(5)	2697.05(2)	2628.74(1)	2697.05(3)	2725.18(4)
Org7	4897.38(5)	4829.49(1)	4872.56(3)	4829.49(2)	4896.32(4)
aveRank	3.91	1.55	3.55	2.00	4.00

(b) MdAE. No significant difference with p-value 0.132.

Data Set	RVM	#_1Dhist	#_rmAll	#_rmPru	#_rmSyn
Maxwell	2176.64(5)	2105.34(3)	2067.03(1)	2113.98(4)	2084.35(2)
Kitchenham	643.28(3)	529.01(1)	697.68(5)	529.01(2)	692.93(4)
Cocomo81	91.94(5)	76.16(1)	76.91(4)	76.27(2)	76.35(3)
Nasa93	100.08(5)	97.39(3)	96.34(2)	97.74(4)	93.70(1)
Org1	574.48(3)	602.31(4)	562.31(2)	605.39(5)	558.05(1)
Org2	624.81(1)	679.41(2)	698.53(4)	679.47(3)	701.90(5)
Org3	460.62(3)	414.93(1)	482.25(5)	415.46(2)	469.70(4)
Org4	2080.82(5)	1968.64(2)	2021.74(4)	1968.64(1)	2009.22(3)
Org5	2616.50(1)	3120.77(2)	3231.82(5)	3185.10(3)	3218.52(4)
Org6	1478.45(1)	1642.26(3)	1671.11(5)	1654.37(4)	1549.50(2)
Org7	4612.05(4)	4436.62(1)	4690.90(5)	4468.52(2)	4525.38(3)
aveRank	3.27	2.09	3.82	2.91	2.91

(c) LSD. Significant difference detected with p-value 0.008.

Data Set	RVM	#_1Dhist	#_rmAll	#_rmPru	#_rmSyn
Maxwell	0.81(5)	0.68(3)	0.68(2)	0.68(4)	0.68(1)
Kitchenham	0.65(3)	0.64(1)	0.69(4)	0.64(2)	0.69(5)
Cocomo81	1.95(5)	1.62(2)	1.67(3)	1.62(1)	1.67(4)
Nasa93	1.11(5)	0.87(3)	0.87(1)	0.87(4)	0.87(2)
Org1	1.00(3)	0.90(1)	1.01(5)	0.90(2)	1.00(4)
Org2	0.78(3)	0.77(1)	0.81(5)	0.77(2)	0.80(4)
Org3	0.74(1)	0.74(2)	1.78(5)	0.74(3)	1.58(4)
Org4	0.92(5)	0.88(2)	0.88(1)	0.88(3)	0.88(4)
Org5	1.09(5)	0.96(1)	0.99(3)	0.96(2)	1.01(4)
Org6	0.99(4)	0.92(1)	1.05(5)	0.92(2)	0.93(3)
Org7	0.95(1)	0.95(2)	0.96(4)	0.95(3)	0.96(5)
aveRank	3.64	1.73	3.45	2.55	3.64

(d) SA. Significant difference detected with p-value $1.68 * 10^{-7}$.

Data Set	RVM	#_1Dhist	#_rmAll	#_rmPru	#_rmSyn
Maxwell	0.52(5)	0.55(1)	0.54(4)	0.55(2)	0.54(3)
Kitchenham	0.54(3)	0.55(1)	0.16(4)	0.55(2)	0.15(5)
Cocomo81	0.48(5)	0.49(1)	0.48(3)	0.49(2)	0.48(4)
Nasa93	0.58(5)	0.59(2)	0.59(3)	0.59(1)	0.59(4)
Org1	0.54(1)	0.51(2)	0.46(5)	0.50(3)	0.48(4)
Org2	0.40(5)	0.41(2)	0.40(4)	0.41(1)	0.40(3)
Org3	0.55(1)	0.53(2)	-0.15(4)	0.53(3)	-0.41(5)
Org4	0.43(3)	0.43(2)	0.42(4)	0.43(1)	0.42(5)
Org5	0.36(3)	0.45(1)	-0.16(4)	0.44(2)	-0.27(5)
Org6	0.47(4)	0.48(2)	-0.20(5)	0.48(1)	0.48(3)
Org7	0.22(4)	0.24(2)	0.22(5)	0.24(1)	0.22(3)
aveRank	3.55	1.64	4.09	1.73	4.00

These results verify the effectiveness of synthetic displacement and the two components as a whole in improving the point prediction performance of RVM. They also suggest that the synthetic displacement has a more significant impact than Bootstrap pruning for effort point estimation.

9.2.2 Comparisons of Uncertain Estimates. For effort uncertain estimation, we follow the same procedure as in Sec. 8.2 to evaluate the performance of these methods based on the 12 CLs.

Regarding hit rate, we conduct Friedman tests at significance level 0.05 for each of the 12 CLs. The null hypothesis (H0) states that the hit rates of the methods are equivalent across data sets. The alternative hypothesis (H1) states that at least one pair of the methods differs in terms of hit rates for this CL. None of the 12 Friedman tests can reject H0 with the p-values ranging from 0.0071 to 0.8945, indicating that #_rmAll, #_rmPru, and #_rmSyn produce similar hit rates to those of RVM and SynB-RVM_1Dhist.

Regarding relative width, we need to compare the width of PIs with similar hit rates following the same evaluation procedures of Sec. 8.2.2. Table 15 lists the 11 benchmark hit rates and the closest actual hit rates of the investigated methods to their corresponding benchmark values over all data sets. Friedman test with the significance level 0.05 does not find significant difference on these hit rates. The p-value of the statistical test is 0.8077 indicating the similarity of these values as desired for a fair comparison of their widths.

The relative widths in line with these similar hit rates are listed in table 16. Friedman test with significance level 0.05 rejects null hypothesis (H0) with the p-value 3.46×10^{-4}, where the alternative hypothesis (H1) is accepted that at least one pair of the methods differs. Post-hoc tests with Holm-Bonferroni corrections by comparing each method against SynB-RVM_1Dhist detect significant difference over RVM, #_rmAll and #_rmSyn with p-value 3.74×10^{-4}, 2.28×10^{-4}, and 2.161×10^{-2} respectively. No significant difference has been found over #_rmPru.

Table 15. The similar hit rates of the investigated methods. The reported values are the mean of 30 runs of 10-fold CV. *B_HitR* denotes the benchmark hit rate. The chosen hit rates may correspond to different CLs.

Data Set	*B_HitR*	RVM	#_1Dhist	#_rmAll	#_rmPru	#_rmSyn
Maxwell	0.7237	0.7538	0.7161	0.7274	0.7161	0.7237
Kitchenham	0.9664	0.9749	0.9664	0.9630	0.9667	0.9621
Cocomo81	0.4392	0.4545	0.4698	0.4392	0.4646	0.4397
Nasa93	0.7993	0.7971	0.8032	0.8014	0.8032	0.7993
Org1	0.9272	0.9289	0.9325	0.9272	0.9316	0.9307
Org2	0.7677	0.7615	0.7677	0.7781	0.7677	0.7677
Org3	0.9765	0.9823	0.9739	0.9765	0.9739	0.9770
Org4	0.9339	0.9489	0.9347	0.9344	0.9347	0.9339
Org5	0.9286	0.9302	0.9571	0.9349	0.9032	0.9286
Org6	0.9636	0.9636	0.9530	0.9636	0.9530	0.9591
Org7	0.9492	0.9111	0.9492	0.9698	0.9492	0.9603

Table 16. The relative widths with similar hit rates of the investigated methods. The reported values are the mean of 30 runs of 10-fold CV. The last row lists the average ranks where significant difference of Friedman tests across all data sets is highlighted in yellow (light grey).

Data Set	RVM	#_1Dhist	#_rmAll	#_rmPru	#_rmSyn
Maxwell	5.3690	1.1964	1.2165	1.2039	1.1998
Kitchenham	4.2984	4.1532	11.4185	4.0922	7.7285
Cocomo81	3.5037	1.0393	0.9260	1.0462	0.9203
Nasa93	7.6685	2.4906	2.5251	2.5062	2.5054
Org1	6.5292	4.2131	4.8810	4.2311	4.8688
Org2	1.7322	1.2572	1.5419	1.2651	1.4916
Org3	8.2039	6.9896	29.4600	7.0333	29.4159
Org4	4.5481	3.5339	3.2107	3.5560	3.2071
Org5	4.7894	3.3688	23.2198	2.4735	17.8844
Org6	11.2810	5.0420	13.7238	5.0735	11.8890
Org7	2.8612	3.0682	7.3683	3.0874	7.8995
aveRank	3.91	1.64	4.00	2.45	3.00

Overall, our experimental results and statistical analyses verify the effectiveness of synthetic displacement and the two components as a whole in improving the uncertain prediction performance of RVM. They also suggest that the synthetic displacement has a more significant impact than Bootstrap pruning for effort uncertain prediction, being consistent with the conclusions on point estimation.

9.2.3 The Parameter of Bootstrap Pruning. It has been shown that pruning has less impact than synthetic displacement in enhancing the baseline performance of RVM. This subsection further investigates how the parameter choices of Bootstrap pruning affect point/uncertain performance. This analysis can also provide information on whether this component should be removed if practitioners have no time to tune its value.

Among the tuning parameters of SynB-RVM (shown in table 7), we find 3 pairs of (M, ρ), each corresponding to one of the pruning rates $\{\tau_0, \tau_{0.1}, \tau_{0.2}\}$. Specifically, we find the best parameter setting of (M, ρ) for τ_0 and the worst settings of (M, ρ) for $\tau_{0.1}$ and $\tau_{0.2}$ in terms of MAE, MdAE, LSD and SA respectively. In this manner, we can compare the best performance without pruning against the worst performance with pruning. Then, their performance is compared across data sets to investigate the impact of the pruning rate. The null hypothesis (H0) states that their performance are equivalent across data sets in terms of point/uncertain prediction.

Regarding point prediction performance, Friedman tests with the significance level 0.05 across data sets reject H0 with the p-values 5.31×10^{-9}, 8.78×10^{-7}, 9.42×10^{-8} and 5.31×10^{-9} for MAE, MdAE, LSD and SA respectively. Post-hoc tests with Holm-Bonferroni corrections using the best performance without pruning as the control group detect significant superiority over the worst performance with pruning rates $\tau_{0.1}$ and $\tau_{0.2}$ in terms of all performance metrics. Effect size across 30 runs for each data set against the control method are large/medium in all data sets except for Maxwell and Org6 in terms of MAE. This demonstrates consistent superiority of using the Bootstrap pruning technique. The magnitude of performance difference in terms of SA varies depending on the data sets. It is usually larger for ISBSG data sets than for SEACRAFT. For instance, the values of SA in Maxwell are similar being all around 0.54, but they have larger difference for Org7 with values of 0.2380, -0.6479, and -0.6132 for τ_0, $\tau_{0.1}$, and $\tau_{0.2}$ respectively. These results indicate that the choice of Bootstrap pruning parameter is important for point prediction. Given bad parameter settings of (M, ρ), using pruning may result in worse performance than not using it.

Regarding hit rate, we conduct the same evaluation procedures as in Sec. 8.2.1. In terms of all performance metrics, Friedman tests with the significance level 0.05 across data sets cannot reject H0 for all CLs with p-values from between 0.0518 to 1. These results indicate that the choice of Bootstrap pruning parameter is insignificant for hit rate. Regarding relative width, Friedman tests with the significance level 0.05 across data sets cannot reject H0 with the p-value 0.3209, 0.2421, 0.0719, and 0.1812 for MAE, MdAE, LSD and SA respectively. These results indicate that choice of Bootstrap pruning parameter is insignificant for relative width.

In summary, our analyses show that parameter choice of Bootstrap pruning has significant effect on point performance, and bad parameter settings of using pruning can lead to worse results than not using it. On the contrary, the parameter choice of pruning does not impact uncertain performance significantly, i.e., hit rate and relative width are robust to the values of the pruning rate. Therefore, practitioners are suggested not to adopt pruning when they do not have time to tune this parameter in case of bad point performance.

9.3 More Comparisons over Bagging Ensemble for Point Prediction

Considering that the differences between SynB-RVM_SpMn and Bag-RVM for point prediction are *synthetic project generation* and *Bootstrap pruning*, we can compare the two methods to judge the effectiveness of the two components as a whole. Wilcoxon sign-rank test [88] is recommended to compare two methods across multiple data sets [19]. The null hypothesis (H0) states that SynB-RVM_SpMn and Bag-RVM are equivalent. The alternative hypothesis (H1) states that they differ. Wilcoxon sign-rank test with the significance level of 0.05 rejects H0 with the p-value 0.0098, 0.0420, 0.000977 and 0.0049 in terms of MAE, MdAE, LSD and SA respectively, verifying the effectiveness of *synthetic displacement* and *Bootstrap pruning* together in producing better point performance, which is consistent with the conclusion by comparing SynB-RVM_SpMn and #_rmAll from Friedman post-hoc tests in Sec. 9.2.1.

In addition, we conduct statistical tests between RVM and Bag-RVM to find whether Bagging ensemble is sufficient to improve the point prediction performance of RVM. The null hypothesis (H0) states that the two methods are equivalent. The alternative hypothesis (H1) states that the two methods are different. Wilcoxon sign-rank test with the significance level 0.05 cannot reject H0 with the p-value 0.1475, 0.2061, 0.6377 and 0.0537 for MAE, MdAE, LSD and SA respectively, indicating that Bagging ensemble cannot promote essential point performance. It is probably because of the invertibility problem when training RVM with replicated projects.

Overall, *synthetic displacement* and *Bootstrap pruning* of SynB-RVM have the merits in improving the point prediction performance, but Bagging ensemble alone cannot.

Table 17. Data columns of point prediction errors and relative widths for all data sets and uncertain methods. 'PF' is the acronym of performance in terms of MAE, MdAE, LSD or SA. There are four such tables to compute Spearman correlations, one for each performance metric.

Group 1. Point Performance	Group 2. relative width
PF of Btstrp-ATLM in Maxwell	rWidth of Btstrp-ATLM in Maxwell
⋮	⋮
PF of Btstrp-ATLM in Org7	rWidth of Btstrp-ATLM in Org7
PF of Btstrp-RVM in Maxwell	rWidth of Btstrp-RVM in Maxwell
⋮	⋮
PF of Btstrp-RVM in Org7	rWidth of Btstrp-RVM in Org7
PF of Emp-ATLM in Maxwell	rWidth of Emp-ATLM in Maxwell
⋮	⋮
PF of Emp-ATLM in Org7	rWidth of Emp-ATLM in Org7
PF of Emp-RVM in Maxwell	rWidth of Emp-RVM in Maxwell
⋮	⋮
PF of Emp-RVM in Org7	rWidth of Emp-RVM in Org7
PF of SynB-RVM_SpMn in Maxwell	rWidth of SynB-RVM_SpMn in Maxwell
⋮	⋮
PF of SynB-RVM_SpMn in Org7	rWidth of SynB-RVM_SpMn in Org7
PF of SynB-RVM_1Dhist in Maxwell	rWidth of SynB-RVM_1Dhist in Maxwell
⋮	⋮
PF of SynB-RVM_1Dhist in Org7	rWidth of SynB-RVM_1Dhist in Org7
PF of SynB-RVM_2Dhist in Maxwell	rWidth of SynB-RVM_2Dhist in Maxwell
⋮	⋮
PF of SynB-RVM_2Dhist in Org7	rWidth of SynB-RVM_2Dhist in Org7

9.4 Correlation of Point Performance and Relative Width for SynB-RVM

This section aims to investigate how much of the good relative width of SynB-RVM is contributed by good point prediction by making use of Spearman correlation between the two factors.

Spearman's rank correlation $r_s \in [-1, +1]$ is a non-parametric statistic that assesses how well the relationship between two variables can be described using a monotonic function [67]. The value +1/-1 means a perfectly increasing/decreasing monotone of one variable over the other. Conventionally, the correlation strength can be interpreted according to $|r_s|$ as [67]: 0.00 – 0.19: very weak, 0.20 – 0.39: weak, 0.40 – 0.59: moderate, 0.60 – 0.79: strong, and 0.80 – 1.00: very strong.

Specifically, we compute Spearman correlation between point prediction errors and relative widths of all uncertain methods across all data sets as illustrated in table 17. The left column consists of the point prediction performance of the uncertain methods across data sets in terms of MAE, MdAE, LSD or SA, and the right column consists of the relative width. For each performance metric, the relative widths are decided following the procedures in Sec. 8.2.2. Altogether, we have four groups of data columns as table 17, each corresponding to one point prediction metric.

Table 18 shows the results. We can see that the correlation is very weak in terms of MAE, MdAE and SA, where good point estimates have little effect on narrower PIs. Whereas, there is *weak* correlation in terms of LSD, where better point estimates lead to narrower PIs. Therefore, the point estimator could sometimes have an influence. As a result, we should choose a good point estimator for use with the uncertain method. Nevertheless, the uncertain method plays a more important role in contributing towards narrower PIs, since the correlation between point performance and relative width is *(very) weak*. The choice of uncertain method also has an impact, as shown in table 11. In particular, some uncertain methods do better than the others.

Table 18. Spearman correlation between point performance and relative widths across all data sets and uncertain methods. The data columns for Spearman calculation is illustrated in table 17.

PF metric	r_s
MAE	-0.005
MdAE	-0.091
LSD	0.297
SA	0.007

Table 19. Summary of the studies on SynB-RVM components. The point prediction metrics include MAE, MdAE, LSD and SA, and the uncertain prediction metrics include hit rate and relative width. Equality/positive sign denotes no-different/significantly better performance of SynB-RVM_SpMn against each method/variant.

PF Metric	RVM	#_rmAll	#_rmPru	#_rmSyn
MAE	+	+	=	+
MdAE	=	=	=	=
LSD	+	+	=	+
SA	+	+	=	+
hitR	=	=	=	=
rWidth	+	+	=	+

9.5 Brief Summary

The three versions of SynB-RVM are similar in terms of both point and uncertain effort prediction, showing that the three methods for deriving final probabilistic prediction in Sec. 5.3 are similar.

Table 19 summarizes the performance investigation of SynB-RVM components. Synthetic displacement and the two components as a whole have the merits in improving the performance of RVM in terms of both point and uncertain prediction. The synthetic displacement has a more significant impact than Bootstrap pruning for both point and uncertain prediction.

10 IMPLICATIONS TO PRACTICE

10.1 Predictive Performance and Data Set Characteristics

This section investigates the correlation between *improvement ratio* of point and uncertain prediction and SEE data characteristics including *complexity*, *linearity*, *clustering* defined as follow.

Improvement ratio of method P_1 over P_2 in terms of performance metric γ is defined as

$$imp\_ratio = \begin{cases} \frac{\gamma(P_2)-\gamma(P_1)}{min\{\gamma(P_1),\gamma(P_2)\}}, & \gamma \in \{\text{relative width, MAE, MdAE, LSD}\} \\ \frac{\gamma(P_1)-\gamma(P_2)}{min\{\gamma(P_1),\gamma(P_2)\}}, & \gamma \in \{\text{SA}\} \end{cases} \tag{15}$$

The improvement ratio of P_1 over P_2 is positive if P_1 is superior to P_2, and negative if otherwise.

Complexity of a data set is defined as the division of the number of features over the number of data samples as

$$complexity = \frac{\#fea}{\#data}. \tag{16}$$

Larger values mean that the data set is harder to estimate the efforts.

Linearity of a data set is defined by the Pearson correlation between effort values in the logarithm scale and size-related features (i.e. line of code or functional size) in the logarithm scale. Logarithm scale is applied because the size-related features and effort values of SEE data is often skewed and thus appropriate transformation, such as logarithm, are often required to form a proper and normal shape [44]. The size-related feature is chosen due to the well-known fact that they are usually the most correlated with the effort on the data sets [16, 53].

Table 20. Analyses of the correlation between prediction performance and data characteristics.

(a) Characteristics of SEE data sets with respect to complexity/linearity/clustering and improvement ratios of SynB-RVM_1Dhist over RVM/EmpATLM quantified by Eq. (15).

Data Set	Complexity	Linearity	#Clusters	Improvement Ratio over RVM					Improvement Ratio over EmpATLM				
				rwdth	MAE	MdAE	LSD	SA	rwdth	MAE	MdAE	LSD	SA
Maxwell	0.3710	0.8184	2	3.7643	0.0575	0.0339	0.1915	0.0568	0.2940	0.0304	0.0051	0.1307	0.0318
Kitchenham	0.0207	0.7263	1	0.0372	0.0306	0.2160	0.0032	0.0167	-0.1538	-0.0443	0.1090	0.0253	-0.0406
Cocomo81	0.2698	0.8466	2	3.4993	0.0285	0.2071	0.2002	0.0240	-0.9902	-1.1752	-1.2891	-1.9609	-0.5541
Nasa93	0.1828	0.8435	3	8.8860	0.0434	0.0276	0.2786	0.0253	-0.7137	-0.2552	-0.4587	-0.2126	-0.1424
Org1	0.0395	0.7422	4	0.5504	-0.0664	-0.0484	0.1199	-0.0687	-0.0582	-0.1616	-0.2329	-0.0238	-0.1386
Org2	0.0938	0.6763	4	0.5631	0.0231	-0.0874	0.0154	0.0393	0.0109	0.0172	0.1735	-0.1290	0.0293
Org3	0.0185	0.8019	2	-0.0168	-0.0459	0.1101	-0.0061	-0.0429	-1.7163	-0.1045	-0.0227	0.0256	-0.0879
Org4	0.0246	0.6263	4	0.2869	0.0071	0.0570	0.0415	0.0067	0.1111	0.0398	-0.2510	-0.0493	0.0552
Org5	0.1429	0.8236	3	0.4375	0.1162	-0.1927	0.1417	0.2509	0.4768	-0.4063	-0.5018	-0.1684	-0.1267
Org6	0.1364	0.7845	1	1.5207	0.0163	-0.1108	0.0710	0.0148	-0.5763	0.7542	-0.4641	-0.1286	4.3986
Org7	0.1429	0.5138	2	0.2805	0.0141	0.0395	-0.0027	0.0660	0.9841	-0.1029	-0.5507	0.0743	-0.2784

(b) Spearman correlation between the improvement ratio and complexity/linearity/clustering across data sets. The moderate/(very) strong correlation strength is highlighted in yellow (light grey), whose interpretation is in section 9.4.

Metric	SynB-RVM_1Dhist vs RVM			SynB-RVM_1Dhist vs EmpATLM		
	Complexity	Linearity	Clustering	Complexity	Linearity	Clustering
rwdth	0.7927	0.5727	0.1180	0.2323	-0.4818	0.2691
MAE	0.6241	0.5364	-0.2171	-0.2642	-0.6000	-0.1605
MdAE	-0.2460	-0.0727	-0.3540	0.1454	-0.3091	0.1416
LSD	0.7563	0.7455	0.1794	-0.2597	-0.5091	-0.2785
SA	0.6606	0.1273	-0.0566	-0.3235	-0.3273	-0.0755

Clustering of a data set is defined by the number of clusters the projects could be divided into. We adopt k-Means [69], for its popularity and effectiveness to improve the performance in the SEE context [60, 75] based on normalised features and Euclidean distance. The cluster number k is determined among $k = \{2, 3, 4, 5\}$ based on the criterion *silhouette values* [70]. The validating procedures are implemented by a MATLAB built-in function *evalclusters()*. Clusters with less than three projects are not considered as a valid division, and thus not counted towards the total number of clusters of the data set. As k-Means is not a deterministic method, i.e., it cannot always retrieve the same clusters when the same data projects are used, we run ten times of the validating procedures and choose the k with the largest silhouette as the final clustering.

Specifically, we calculate the improvement ratio of SynB-RVM_1Dhist over RVM/EmpATLM on each data set, and then compute the Spearman correlation between these improvement ratios and complexity/linearity/clustering characteristics across data sets. SynB-RVM_1Dhist is chosen among the three versions for usually performing the best, RVM is chosen for being the baseline of SynB-RVM, and EmpATLM is chosen for performing the best among the ATLM-based methods. Table 20(a) lists the characteristics of data sets and the improvement ratios of SynB-RVM over RVM/EmpATLM. Some data sets such as Maxwell and Cocomo81 are more linear than others such as Org7 and Org4. The complexity of the data sets varies ranging from Org3 at 0.0185 (relatively easy) to Maxwell at 0.3710 (relatively hard). The number of clusters is usually not more than four. Overall, we have multiple types of data sets to represent the SEE task in practice.

Table 20(b) lists the Spearman correlations between the improvement ratios and the characteristics of data sets. With respect to RVM, the improvement ratios of our method are larger in the data sets that are more complex or more linear in terms of both point and uncertain predictions, as illustrated by the large positive correlations highlighted in yellow (light grey) in the table. Regarding complexity, a possible reason for these results is that by using Bootstrap resampling, our method would enlarge the training set of this Bootstrap bag, and thus obtain larger improvement over the prediction performance of the baseline RVM on more complex data sets. Regarding linearity, a

possible reason could be the synthetic displacement technique of our method. The synthetic project is generated by a linear combination of a replicated project and its furthest neighbour. Thus, a more linear data set would lead to synthetic projects with higher quality, potentially contributing to better prediction performance. With respect to EmpATLM, the improvement ratios of SynB-RVM are smaller in more linear data sets in terms of both point and uncertain predictions. A possible reason is that EmpATLM is designed (and thus should be more suitable) for more linear data sets, and such factor is stronger than the enhancement of more linear data to SynB-RVM. With respect to clustering, the correlations are (very) weak and thus neglectable for the improve ratios over both RVM and EmpATLM. It means that the clustering of a data set does not have impact on the choice of adopting our method against other uncertain approaches.

Overall, when the data sets are harder or more linear, practitioners are suggested to use SynB-RVM over RVM for better point and uncertain prediction. The more linear the data set, the smaller the improvement ratio of SynB-RVM over EmpATLM.

10.2 Understandability vs. Better Performance

This section aims to discuss the trade-off between understandability and better performance of the uncertain methods investigated.

ATLM is easy to understand as a variant of multiple linear regression; whereas RVM lays its foundation on Bayesian framework and thus requires more background knowledge to understand. As our proposed methods are based on RVM, the model is more difficult to interpret than those based on ATLM. Nevertheless, the mechanisms used to produce the PIs of the uncertain approaches have conceptually equal understandability: EmpSEEr[4] employs the error distribution of training projects to decide the upper and lower bounds of the PI of a testing project; BtstrpSEEr[5] extracts two values from multiple point predictions to form the upper and lower bounds of the PI; SynB-RVM integrates multiple uncertain predictions into one through averaging. Their final point prediction is either a single value (for EmpSEEr) or the mean of multiple values (for BtstrpSEEr and SynB-RVM).

In practice, when practitioners are more keen to understand the underlying estimation models, ATLM-related methods would be more attractive, with a statistical sacrifice on the uncertain prediction performance. In particular, if the practitioners do not value PIs, ATLM would provide both interpretability and good point estimates, being recommended. However, when the practitioners are more concerned on better uncertain estimation, SynB-RVM would be their best option for being robust and having statistically similar or better point and uncertain prediction. Overall, it is a trade-off between superiority in prediction performance and understandability when selecting an SEE method.

10.3 Time Complexity of the Uncertain Methods

This section aims to analyse the time complexity of the uncertain methods with respect to training and testing phases given training set size n. Since the uncertain methods are based on ATLM or RVM, we would analyse the complexity of RVM and ATLM as follows.

ATLM is a variant of multivariate linear regression with extra data transformation (Sec. 2.4), so the time complexity of ATLM is $\mathbb{O}(n^3)$ for model training and $\mathbb{O}(n)$ for effort estimation of a new project, ignoring the time complexity of data preprocessing. Note that ATLM itself cannot provide uncertain prediction and needs to be integrated with EmpSEEr/BtstrpSEEr.

The model parameters of RVM need to be updated alternatively [81]. Suppose K to be the iterations by which the learning algorithm of RVM converges. At each iteration, the training

[4]EmpSEEr denotes EmpRVM or EmpATLM.
[5]BtstrpSEEr denotes BtstrpRVM or BtstrpATLM.

Table 21. Time complexity of the uncertain methods with respect to training and predicting phases. Denote n as the number of training samples, M as the number of Bootstrap bags, and K as the iterations that the learning algorithm of RVM converges. Conventionally, M is smaller than n. Note that ATLM itself cannot provide uncertain predictions. In practice, the training and prediction processes of BtstrpSEEr/SynB-RVM can be proceeded in parallel, leading to much reduced time complexity.

Complexity	ATLM	RVM	BtstrpATLM	BtstrpRVM	EmpATLM	EmpRVM	SynB-RVM
training	$\mathbb{O}(n^3)$	$\mathbb{O}(Kn^3)$	$\mathbb{O}(M*n^3)$	$\mathbb{O}(M*Kn^3)$	$\mathbb{O}(n^3)$	$\mathbb{O}(Kn^3)$	$\mathbb{O}(M*Kn^3)$
prediction	$\mathbb{O}(n)$	$\mathbb{O}(n^2)$	$\mathbb{O}(M*n)+\mathbb{O}(M^2)$	$\mathbb{O}(M*n^2)+\mathbb{O}(M^2)$	$\mathbb{O}(n)$	$\mathbb{O}(n^2)$	$\mathbb{O}(M*n^2)$

process includes an $n \times n$ matrix multiplication and a $n \times n$ matrix inversion, leading to the overall complexity $\mathbb{O}(Kn^3)$. The testing phase includes the multiplication of an $n \times 1$ vector and an $n \times n$ matrix, leading to the complexity $\mathbb{O}(n^2)$.

BtstrpSEEr's training phase consists of constructing M base models based on the M Bootstrap bags leading to the complexity $\mathbb{O}(M*n^3)$ for BtstrpATLM and $\mathbb{O}(M*Kn^3)$ for BtstrpRVM (Sec. 7.4.2). BtstrpSEEr's prediction phase consists of: (1) predicting M point estimates for the testing project, each corresponding to one base model, (2) sorting the M point estimates and (3) extracting certain percentiles for upper and lower bounds of the PI. Accordingly, the time complexity of the prediction phase is $\mathbb{O}(M*V)+\mathbb{O}(M^2)+\mathbb{O}(1)$, where V denotes the prediction complexity of the base model and $\mathbb{O}(M^2)$ is the worst complexity of sorting M variables. In particular, the prediction time complexity of BtstrpATLM is $\mathbb{O}(M*n)+\mathbb{O}(M^2)$, and that of BtstrpRVM is $\mathbb{O}(M*n^2)+\mathbb{O}(M^2)$.

EmpSEEr's training phase consists of (1) constructing the model, (2) computing the training errors, (3) sorting the error values of size n, and (4) extracting certain percentiles of the error values for upper and lower bounds of the PI (Sec. 7.4.3). Accordingly, the time complexity is $\mathbb{O}(V)+\mathbb{O}(n)+\mathbb{O}(n^2)+\mathbb{O}(1)$, where V denotes the training complexity of the base model and $\mathbb{O}(n^2)$ is the worst time complexity of sorting n variables. In particular, the training time complexity is $\mathbb{O}(n^3)$ for EmpATLM and $\mathbb{O}(Kn^3)$ for EmpRVM. EmpSEEr's prediction phase consists of (1) predicting the point estimate of the testing project, and (2) computing the lower and upper bounds of PIs as Eq. (12). Accordingly, the time complexity is $\mathbb{O}(V)+\mathbb{O}(1)$. In particular, the prediction time complexity is $\mathbb{O}(n)$ for EmpATLM and $\mathbb{O}(n^2)$ for EmpRVM.

SynB-RVM's training phase consists of constructing M RVMs from the M Bootstrap bags (Sec. 4), leading to the complexity $\mathbb{O}(M*Kn^3)$. SynB-RVM's prediction phase consists of (1) obtaining M uncertain estimates of the testing project, each corresponding to one of the RVMs, (2) combining the M uncertain estimates using Eq. (4)~(6), and (3) extracting certain percentiles for upper and lower bounds of PIs (Sec. 5), leading to the complexity $\mathbb{O}(M*n^2)+\mathbb{O}(M)+\mathbb{O}(1)=\mathbb{O}(M*n^2)$.

Table 21 summarizes the time complexity of the uncertain methods. We can see that EmpATLM, EmpRVM and RVM have lower time complexity in both training and predicting phases compared to SynB-RVM. Nevertheless, SynB-RVM can achieve significantly better point/uncertain prediction performance as shown in table 13. In practice, the time complexity of SynB-RVM can be largely reduced by proceeding the training and predicting with M Bootstrap ensembles in parallel.

Altogether, there is always a trade-off between the cost and good performance with respect to uncertain prediction: the practitioners are suggested to take faster methods such as EmpATLM, when they are more concerned with the computational efficiency; whereas, they are suggested to take SynB-RVM, when they are keen for better uncertain prediction.

11 FURTHER DISCUSSION

SynB-RVM's base model: SynB-RVM needs the base models, like RVMs, to be capable of deriving probabilistic effort estimations themselves. Thus, its aim is to combine those uncertain predictions into a unified one when making a prediction for a testing project. In this way, the prediction

uncertainty comes from each of the base models when giving uncertain predictions for the testing project. In contrast, the other PI methods utilize only the point estimates from their base models, based on which their prediction uncertainty for the testing project is derived. Therefore, our method can be considered to use richer uncertain information in order to produce its PIs than the other PI methods. Moreover, our method could be considered as a way to combine potentially 'weaker' uncertain predictions into a stronger one. The experimental results in comparing the uncertain prediction between RVM and SynB-RVM (Sec. 8.2) verify this statement.

SynB-RVM's extension: SynB-RVM can be extended with other base models as long as they can provide probabilistic predictions for a testing project. The base model can be a single predictor like RVM that can provide uncertain prediction itself or an ensemble of point estimators like EmpRVM that can produce uncertain prediction as a group. In a sense, a more general name of the proposed method can be *Synthetic Bootstrap ensemble of Probabilistic Predictors*. The experimental investigation and evaluation on the more general framework are left as future work.

Non-symmetric uncertain effort prediction: SynB-RVM assumes that the prediction uncertainty for a project effort follows Gaussian distribution by considering the noises such as the mistake when collecting the actual efforts and the limitation of SEE model as independent random processes (see Sec. 3.2). The Gaussian assumption is reasonable according to central limit theorem (CLT). However, it disregards the fact that the software effort has to be positive.

For Gaussian effort uncertainty, the probabilistic prediction is symmetric as shown in Fig. 1. However, when the predicted probabilistic prediction overlaps the negative quadrant, the positive requirement of project efforts would push the negative values right-moved to the non-negative values, making the symmetric probabilistic prediction right-skewed. Previous empirical observations were in line with the above description stating that the distribution of uncertainty for SEE was not symmetric but usually had a longer right tail [43, 49].

In this case, non-symmetric right-skewed PIs of CLs would be better, for which the right parts of PIs are longer and heavier than the left one. As a result, the Gaussian effort noise assumption may have problem in causing less informative PIs when the probabilistic effort predictions overlap the negative quadrant since the symmetric PIs are produced. To cater such problems, we could relax the Gaussian assumption on the effort noise in the model of RVM by considering other non-symmetric effort noise models such as righted-skewed Gaussian [6] or Gamma distribution [33]. However, the revision on the noise assumption of RVM model would fail the analytical solutions, and lead to considerably complicated deductions. One potential simpler solution is to adjust the predicted effort probability slightly rightward according to the skewness of the point estimates of the RVMs. We leave it as our future study for further improving our method's uncertain predictions.

12 THREATS TO VALIDITY

A potential threat to validity is that to answer RQ3 we did multiple statistical tests possibly inducing type I error. For instance, besides Friedman post-hoc tests with Holm-Bonferroni corrections, we did Wilcoxon sign-rank test between Bag-RVM and SynB-RVM_SpMn to study the effectiveness of synthetic displacement and Bootstrap pruning. However, we do not consider it to be very serious threat to this study, because these p-values were considerably small indicating strong difference or very large indicating strong identity.

Another potential threat to validity is the three extra parameters of SynB-RVM compared with RVM. We did not investigate a large number of values for these parameters, i.e., only three values were investigated for each model parameter as shown in table 7. Despite that, our method showed significantly better point estimates and much effective PIs compared with other RVM-related

methods. Therefore, we do not consider further parameter tuning as essential for this study. As a future work, we will investigate the impact of parameter settings further.

This study has not explored a full range of SEE methods as base models of our proposed method. More empirical experiments are required to repeat this study using other base models than RVM and ATLM into the proposed method that can provide uncertain effort prediction. Nevertheless, given the choice of base models that have been shown to be competitive with the state-of-the-art effort estimators [79, 87], this paper offers much support in providing more informative and meaningful uncertain effort estimations.

13 CONCLUSIONS

We propose a novel SEE method called *synthetic Bootstrap ensemble of RVMs* (SynB-RVM) designed for better PIs with CLs. SynB-RVM adopts Bootstrap resampling to produce multiple RVMs using different training bags that are sampled from the original training data with replacement. It then ensembles those RVMs, each of which can provide uncertain prediction for a testing project, into a unified probabilistic effort estimator. Based on them, PIs with CLs can be easily generated. SynB-RVM has three versions, namely *SynB-RVM_SpMn*, *SynB-RVM_1Dhist* and *SynB-RVM_2Dhist*, depending on the ways the final predictions are produced. We validate SynB-RVM by answering three research questions as follows.

RQ1: When used as a point estimator, how well can SynB-RVM perform compared with the state-of-the-art SEE methods? Experiments show that SynB-RVM can either significantly outperform or have similar point estimation performance compared with the investigated methods, showing to be promising in the context of SEE. In particular, SynB-RVM significantly outperforms RVM (EmpRVM), BtstrpRVM (Bag-RVM), *k*NN, SVR, MLP, RT, Bag-RT and Bag-SVR in terms of at least one metric, and performs similar to ATLM (EmpATLM) and BtstrpATLM (Bag-ATLM).

RQ2. When used as an uncertain estimator, can SynB-RVM's PIs achieve adequate hit rates with narrower and more informative intervals? In terms of hit rate, experimental results show that SynB-RVM and RVM can usually achieve significantly better hit rates. The hit rates from SynB-RVM and RVM are similar. In terms of relative width, SynB-RVM can produce significantly better PIs than those from RVM and BtstrpRVM when reaching similar hit rates. Even though the relative widths of SynB-RVM are similar to those produced by EmpRVM, EmpATLM and BtstrpATLM, SynB-RVM can usually achieve much better PIs with medium/large effect size in most data sets. Therefore, SynB-RVM can achieve the overall better uncertain performance.

RQ3. Which components of SynB-RVM contribute to the point and uncertain prediction improvement? In detail, (1) are the three methods for deriving the final probabilistic prediction similar? And (2) do the synthetic displacement and Bootstrap pruning of SynB-RVM improve the final point and uncertain prediction? Statistical studies on the three versions of SynB-RVM show that they are similar in terms of both point and uncertain effort prediction. synthetic displacement and the two components as a whole have the merits in improving the performance of RVM in terms of both point and uncertain prediction, as opposed to the Bootstrap sampling itself. The synthetic displacement has a more significant impact than Bootstrap pruning for better point and uncertain prediction.

Besides the main contribution in proposing and validating SynB-RVM by answering the above research questions, this paper is the first study to provide a thorough experimental comparison on uncertain effort estimation including several types of PI methods (Sec. 3). In practice, due to the capability of providing uncertain effort prediction, the proposed SynB-RVM has the potential to help PMs to make better informed decisions by accessing the project estimation risks. It could also

provide more flexibility to PMs when making decisions to bring more profits for their organizations. Based on the encouraging results obtained in this work, we would like to further evaluate this method both quantitatively and qualitatively in practice, with industry.

The proposed SynB-RVM still has room for improvement such as the non-symmetric effort PIs as discussed in Sec. 11. The investigation of new strategies is left as future work, as well as the experimentation with more base models that can provide probabilistic effort prediction.

ACKNOWLEDGMENTS

This work was supported by National Key R&D Program of China (Grant No. 2017YFC0804003), EPSRC (Grant Nos. EP/J017515/1, EP/R006660/1, and EP/P005578/1), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Peacock Plan (Grant No. KQTD2016112514355531), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284), and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

REFERENCES

[1] E. Alpaydin. 1998. Techniques for Combining Multiple Learners. In *Engineering Intelligent Systems*. 6–12.
[2] L. Angelis and I. Stamelos. 2000. A Simulation Tool for Efficient Analogy Based Cost Estimation. *Empirical Software Engineering (ESE)* 5, 1 (2000), 35–68.
[3] L. Angelis and I. Stamelos. 2002. Reply to Comments by M. Jorgensen on the Paper: 'A Simulation Tool for Efficient Analogy Based Cost Estimation' by L. Angelis and I. Stamelos, Published in Empirical Software Engineering, 5, 35-68 (2000). *Empirical Software Engineering (ESE)* 7, 4 (2002), 377–381.
[4] A. Arcuri and L. Briand. 2011. A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. In *International Conference on Software Engineering (ICSE)*. 1–10.
[5] J. S. Armstrong. 2001. *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Springer.
[6] A. Azzalini. 2013. *The Skew-Normal and Related Families*. Cambridge University Press. https://doi.org/10.1017/CBO9781139248891
[7] A. Bakir, B. Turhan, and A. Bener. 2011. A Comparative Study for Estimating Software Development Effort Intervals. *Software Quality Journal (SQJ)* 19, 3 (2011), 537–552.
[8] B. Baskeles, B. Turhan, and A. Bener. 2007. Software Effort Estimation Using Machine Learning Methods. In *International Symposium on Computer and Information Sciences*. 1–6.
[9] S. Bibi, I. Stamelos, and E. Angelis. 2004. Software Cost Prediction with Predefined Interval Estimates. In *Software Measurement European Forum*. 237–246.
[10] C. Bishop. 2006. *Pattern Recgonition and Machine Learning*. Springer.
[11] Barry W. Boehm. 1981. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ.
[12] P. Braga and A. Oliveira. 2007. Software Effort Estimation Using Machine Learning Techniques with Robust Confidence Intervals. In *International Conference on Tools with Artificial Intelligence*. 181–185.
[13] L. Breiman. 1996. Bagging predictors. *Journal of Machine Learning* 24, 2 (1996), 123–140.
[14] L. C. Briand, K. Emam, and F. Bomarius. 1998. CoBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment. In *International Conference on Software Engineering (ICSE)*. 390–399.
[15] M. H. Cartwright, M. J. Shepperd, and Q. Song. 2003. Dealing with Missing Software Project Data. In *International Workshop on Enterprise Networking and Computing in Healthcare Industry*. 154–165.
[16] Zhihao Chen, Tim Menzies, Dan Port, and Barry Boehm. 2005. Feature Subset Selection Can Improve Software Cost Estimation Accuracy. In *International Conference on Predictor Models in Software Engineering (PROMISE)*. 1–6.
[17] V. Cherkassky and Y. Ma. 2004. Practical selection of SVM Parameters and Noise Estimation for SVM Regression. *Neural Networks* 17, 1 (2004), 113 – 126.
[18] S. Chulani, B. Boehm, and B. Steece. 1999. Bayesian Analysis of Empirical Software Engineering Cost Models. *IEEE Transactions on Software Engineering (TSE)* 25, 4 (1999), 573–583.
[19] J. Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research (JMLR)* 7 (2006), 1–30.
[20] Kitchenham doi. 2017. https://doi.org/10.5281/zenodo.268457. (2017).
[21] Maxwell doi. 2009. https://doi.org/10.5281/zenodo.268461. (2009).
[22] Nasa93 doi. 2008. https://doi.org/10.5281/zenodo.268419. (2008).

[23] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. 1996. Support Vector Regression Machines. In *Neural Information Processing Systems (NIPS)*. 155–161.
[24] B. Efron. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* 7, 1 (1979), 1–26.
[25] B. Efron and R. Tibshirani. 1993. *An Introduction to the Bootstrap*. New York: Chapman and Hall.
[26] A. Faul and M. Tipping. 2001. Analysis of Sparse Bayesian Learning. In *Advances in Neural Information Processing Systems 14*. MIT Press, 383–389.
[27] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. 2003. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Transactions on Software Engineering (TSE)* 29 (2003), 985–995.
[28] M. Harman, F. Ferruci, and F. Sarro. 2014. Search-Based Software Project Management. In *Software Project Management in a Changing World*, Gunther Ruhe and Claes Wholin (Eds.). Springer.
[29] T. Hastie, R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*. Springer.
[30] A. Idri, M. Hosni, and A. Abran. 2016. Systematic Literature Review of Ensemble Effort Estimation. *Journal of Systems and Software (JSS)* 118 (2016), 151 – 175.
[31] ISBSG. 2011. The International Software Benchmarking Standards Group. http://www.isbsg.org. (2011).
[32] D. N. Joanes and C. A. Gill. 1998. Comparing Measures of Sample Skewness and Kurtosis. *Journal of the Royal Statistical Society. Series D (The Statistician)* 47, 1 (1998), 183–189.
[33] N. L. Johnson and S. Kotz. 1970. Distributions in Statistics: Continuous Univariate Distributions. Vol. 1,2. Boston etc.: Houghton Mifflin Company. I: XIV, 300 p.; II: XIII, 306. (1970).
[34] M. Jørgensen. 2002. Comments on 'A Simulation Tool for Efficient Analogy Based Cost Estimation', by L. Angelis and I. Stamelos, Published in Empirical Software Engineering, 5, 35-68 (2000). *Empirical Software Engineering (ESE)* 7, 4 (2002), 375–376.
[35] M. Jørgensen. 2004. Realism in Assessment of Effort Estimation Uncertainty: It Matters How You Ask. *IEEE Transactions on Software Engineering (TSE)* 30, 4 (2004), 209–217.
[36] M. Jørgensen. 2005. Evidence-based Guidelines for Assessment of Software Development Cost Uncertainty. *IEEE Transactions on Software Engineering (TSE)* 32, 11 (2005), 942–954.
[37] M. Jørgensen. 2013. The Influence of Selection Bias on Effort Overruns in Software Development Projects. *Information and Software Technology (IST)* 55, 9 (2013), 1640–1650.
[38] M. Jørgensen, U. Indahl, and D. Sjøberg. 2003. Software Effort Estimation by Analogy and "Regression Toward the Mean". *Journal of Systems and Software (JSS)* 68, 3 (2003), 253–262.
[39] M. Jørgensen and M. Kjetil. 2006. How Large are Software Cost Overruns? A Review of the 1994 CHAOS Report. *Information and Software Technology (IST)* 48, 4 (2006), 297–301.
[40] M. Jørgensen and K. Sjøberg. 2003. An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy. *Information and Software Technology (IST)* 45, 3 (2003), 123 – 136.
[41] M. Jørgensen, K. H. Teigen, and K. Molokken. 2004. Better Sure than Safe? Overconfidence in Judgement based Software Development Effort Prediction Intervals. *Journal of systems and software* 70 (2004), 79–93.
[42] V. Khatibi Bardsiri, D. N. Jawawi, S. Z. Hashim, and E. Khatibi. 2014. A Flexible Method to Estimate the Software Development Effort Based on the Classification of Projects and Localization of Comparisons. *Empirical Software Engineering (ESE)* 19, 4 (2014), 857–884.
[43] B. Kitchenham and S. Linkman. 1997. Estimates, Uncertainty, and Risk. *IEEE Software* 14, 3 (1997), 69–74.
[44] B. Kitchenham and E. Mendes. 2009. Why Comparative Effort Prediction Studies May Be Invalid. In *International Conference on Predictor Models in Software Engineering (PROMISE)*. 4:1–4:5.
[45] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan. 2002. An Empirical Study of Maintenance and Development Estimation Accuracy. *Journal of Systems and Software (JSS)* 64, 1 (2002), 57–77.
[46] B. Kitchenham, L. Pickard, S. Linkman, and P. Jones. 2003. Modeling Software Bidding Risks. *IEEE Transactions on Software Engineering (TSE)* 29, 6 (2003), 542–554.
[47] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd. 2001. What Accuracy Statistics Really Measure. *IEE Proceedings - Software Engineering* 148, 3 (2001), 81–85.
[48] M. Klas, A. Trendowicz, Y. Ishigai, and H. Nakao. 2011. Handling Estimation Uncertainty with Bootstrapping: Empirical Evaluation in the Context of Hybrid Prediction Methods. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 245–254.
[49] M. Klas, A. Trendowicz, A. Wickenkamp, J. Munch, N. Kikuchi, and Y. Ishigai. 2008. The Use of Simulation Techniques for Hybrid Software Cost Estimation and Risk Analysis. In *Advances in computers*. Software Development, Vol. 74. Academic Press, 115–174.
[50] M. Klas, A. Trendowizc, Y. Ishigai, and H. Nakao. 2011. Handling Estimation Uncertainty with Bootstrapping: Empirical Evaluation in the Context of Hybrid Prediction Methods. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 245–254.

[51] E. Kocaguneli and T. Menzies. 2012. Exploiting the Essential Assumptions of Analogy-Based Effort Estimation. *IEEE Transactions on Software Engineering (TSE)* 38, 2 (2012), 425–438.

[52] S. Laqrichi, F. Marmier, D. Gourc, and J. Nevoux. 2015. Integrating Uncertainty in Software Effort Estimation Using Bootstrap based Neural Networks. *IFAC Symposium on Information Control Problems in Manufacturing* 48, 3 (2015), 954–959.

[53] Y. Li, M. Xie, and T. Goh. 2009. A Study of Project Selection and Feature Weighting for Analogy Based Software Cost Estimation. *Journal of Systems and Software (JSS)* 82, 2 (2009), 241 – 252.

[54] K. Maxwell. 2002. *Applied Statistics for Software Managers.* Prentice Hall PTR.

[55] E. Mendes and N. Mosley. 2008. Bayesian Network Models for Web Effort Prediction: A Comparative Study. *IEEE Transactions on Software Engineering (TSE)* 34, 6 (2008), 723–737.

[56] S. Mensah, J. Keung, M. Bosu, and K. Bennin. 2018. Duplex Output Software Effort Estimation Model with Self-Guided Interpretation. *Information and Software Technology* 94 (2018), 1 – 13.

[57] T. Menzies, R. Krishna, and D. Pryor. 2015. The Promise Repository of Empirical Software Engineering Data. http://openscience.us/repo. North Carolina State University, Department of Computer Science. (2015).

[58] T. Menzies, R. Krishna, and D. Pryor. 2017. The SEACRAFT Repository of Empirical Software Engineering Data. https://zenodo.org/communities/seacraft. (2017).

[59] L. L. Minku and X. Yao. 2012. Can Cross-company Data Improve Performance in Software Effort Estimation?. In *International Conference on Predictive Models in Software Engineering (PROMISE)*. ACM, 69–78.

[60] L. L. Minku and X. Yao. 2012. Ensembles and Locality: Insight on Improving Software Effort Estimation. *Information and Software Technology (IST)* 55, 8 (2012), 1512–1528.

[61] L. L. Minku and X. Yao. 2013. Software Effort Estimation as a Multiobjective Learning Problem. *ACM Transactions Software Engineering and Methodology (TOSEM)* 22, 4 (2013), 35:1–35:32.

[62] M. Momma and K. P. Bennett. 2002. A Pattern Search Method for Model Selection of Support Vector Regression. In *International Conference on Data Mining*. 261–274.

[63] I. Myrtveit, E. Stensrud, and M. Shepperd. 2005. Reliability and Validity in Comparative Studies of Software Prediction Models. *IEEE Transactions on Software Engineering (TSE)* 31, 5 (2005), 380–391.

[64] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. 1996. *Applied Linear Statistical Models.* McGraw-Hill.

[65] NIST. 2012. Engineering Statistics, NIST/SEMATECH e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/. (2012).

[66] Adriano L.I. Oliveira. 2006. Estimation of Software Project Effort with Support Vector Regression. *Neurocomputing* 69, 13 (2006), 1749 – 1753.

[67] E. S. Pearson and Barbara A. S. Snow. 1962. Tests for Rank Correlation Coefficients: I. *Biometrika* 49, 1-2 (1962), 185–191.

[68] C. Pendharkar, H. Subramanian, and A. Rodger. 2005. A Probabilistic Model for Predicting Software Development Effort. *IEEE Transactions on Software Engineering (TSE)* 31, 7 (2005), 615 – 624.

[69] L. Rokach and O. Maimon. 2005. *Clustering Methods.* Springer US, Boston, MA, 321–352.

[70] Peter J. Rousseeuw. 1987. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.* 20 (1987), 53 – 65.

[71] F. Sarro, A. Petrozziello, and M. Harman. 2016. Multi-objective Software Effort Estimation. In *International Conference on Software Engineering (ICSE)*. 619–630.

[72] S. M. Satapathy and S. K. Rath. 2014. Use Case Point Approach Based Software Effort Estimation using Various Support Vector Regression Kernel Methods. *CoRR* abs/1401.3069 (2014). http://arxiv.org/abs/1401.3069

[73] P. Sentas, L. Angelis, and I. Stamelos. 2003. Multinomial Logistic Regression Applied on Software Productivity Prediction. In *Panhellenic Conference in Information*.

[74] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris. 2005. Software Productivity and Effort Prediction with Ordinal Regression. *Information and Software Technology (IST)* 47, 1 (2005), 17 – 29.

[75] Y. S. Seo, K. Yoon, and D. H. Bae. 2008. An Empirical Analysis of Software Effort Estimation with Outlier Elimination. In *International Conference on Predictive Models in Software Engineering (PROMISE)*. 25–32.

[76] M. Shepperd and S. McDonell. 2012. Evaluating Prediction Systems in Software Project Estimation. *Information and Software Technology (IST)* 54 (2012), 820–827.

[77] M. Shepperd and C. Schofield. 1997. Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering (TSE)* 23, 12 (1997), 736–743.

[78] L. Song, L. L. Minku, and X. Yao. 2013. The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines. In *International Conference on Predictor Models in Software Engineering (PROMISE)*. Baltimore, USA, 9:1–9:10.

[79] L. Song, L. L. Minku, and X. Yao. 2014. The Potential Benefit of Relevance Vector Machine to Software Effort Estimation. In *International Conference on Predictor Models in Software Engineering (PROMISE)*. Turin, Italy, 52–61.

[80] I. Stamelos, L. Angelis, P. Dimou, and E. Sakellaris. 2003. On the Use of Bayesian Belief Network for Prediction of Software Productivity. *Information and Software Technology (IST)* 45, 1 (2003), 51–60.
[81] M. Tipping. 2001. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning* 1 (2001), 211–244.
[82] V. Vapnik. 1982. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.
[83] V. Vapnik. 1995. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., New York, NY, USA.
[84] V. Vapnik. 1998. *Statistical Learning Theory.* Wiley.
[85] A. Vargha and H. D. Delaney. 2000. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* 25, 2 (6 2000), 101–132.
[86] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang. 2012. Systematic Literature Review of Machine Learning Based Software Development Effort Estimation Models. *Information and Software Technology (IST)* 54, 1 (2012), 41–59.
[87] P. A. Whigham, C. A. Owen, and S. G. Macdonell. 2015. A Baseline Model for Software Effort Estimation. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 24, 3 (2015), 20:1–20:11.
[88] F. Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (1945), 80–83.
[89] B. Wlodzimierz. 1995. *The Normal Distribution: Characterizations with Applications.* Springer-Verlag.
[90] Z. Zhou. 2012. *Ensemble Methods: Foundations and Algorithms* (1st ed.). Chapman & Hall/CRC.